NASA SP-223

THE NASTRAN PROGRAMMER'S MANUAL

Frank J. Douglas

Editor

September 1970

Scientific and Technical Information Division
OFFICE OF TECHNOLOGY UTILIZATION
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
1970
Washington, D.C.

For sale from Computer Software Management and Information Center (COSMIC)
Barrows Hall, University of Georgia, Athens, Georgia 30601 – Price $15.00

FOREWORD

For many decades the art of structural analysis remained essentially stagnant, not because of lack of theoretical understanding but because of limitations on capability for numerical computation. The analyst was constrained to apply approximate continuum solutions of the equations of elasticity to his structure which yield, for the most part, average stress and strain distributions not taking account of localized structural features. Fortunately, most aircraft could be approximated by a collection of beam-like one-dimensional structures--that is, structures where state of deformation could be adequately described by a set of functions of a single space coordinate. But, spacecraft structures and very high speed aircraft began to depart appreciably from such idealizations. Structural analysts began to adapt theory to aircraft structures viewed as an assemblage of a finite number of elastic components over 20 years ago. In one of the first papers on the subject, Levy in 1947 isolated each component, placed it in static equilibrium, and regarded the internal forces as the unknown quantities. The principle of minimum strain energy was applied to determine the correct internal force distribution. Subsequent papers by Schuerch, Levy, Turner, Clough, Martin and others took the deformations of the various discrete components of the structure as the unknown quantities, instead of the forces on the components. The correct deformation pattern was obtained by applying the principle of minimum potential energy. In all of these applications, matrix formulations were developed early as a means of organizing the bookkeeping. Numerical solutions were, for the most part, reduced to the processes of addition, subtraction, multiplication, inversion and finding the characteristic roots and vectors of matrices. But the analyst was limited at first by the desk calculator. The inversion of ten by ten matrices by Crout's or an equivalent method was a formidable exercise. It is no wonder that the earliest users of digital computers in the aircraft companies were the Structures and Dynamics Divisions of the Engineering Department. It was indeed fortunate that the birth of the digital computer came when it did.

But the development of software for structural analysis became a separated and individualistic activity with little or no coordination. There grew a tendency toward proprietary secrecy. Little compatability was developed among the various structural analysis programs throughout the country. The present program, called NASTRAN, is an attempt to bring together the best features of the state

of the art into a single program for the analysis of large complex structures. The result is an exemplary beginning. It will not satisfy every requirement at first. But, it forms a framework for an enormous step forward in the development of a common base. The NASA staff and contractors who conceived of and took part in this important new development are to be commended for their foresight and skill.

Dr. Raymond L. Bisplinghoff
Dean, School of Engineering
Massachusetts Institute of Technology

INTRODUCTION TO THE NASTRAN PROGRAM

The acronym NASTRAN is formed from NAsa STRuctural ANalysis. NASTRAN is a general purpose digital computer program for the analysis of large complex structures and has its origins in the research councils of NASA. During the annual review of NASA's research program in the area of structural dynamics by Douglas Michel of Headquarters in January 1964, it became apparent that there was considerable effort by many of the Centers to develop computer programs for structural analysis, designed to meet each of the Centers' particular needs. It was suggested that perhaps a single program could meet all their needs. The Office of Advanced Research and Technology appointed a committee with representation from eight NASA Centers to study this possibility. Thus formed, the AD HOC GROUP ON COMPUTER METHODS IN STRUCTURAL ANALYSIS was constituted as follows:

Ames Research Center	Richard M. Beam
	Perry P. Polentz
Flight Research Center	Richard J. Rosecrans
Goddard Space Flight Center	Thomas G. Butler
	Peter A. Smidinger
Jet Propulsion Laboratory	Marshall E. Alper
	Robert M. Bamford
Langley Research Center	Herbert J. Cunningham
Lewis Research Center	William C. Scott
	James D. McAleese
Manned Spacecraft Center	C. Thomas Modlin, Jr.
	William W. Renegar
Marshall Space Flight Center	Robert L. McComas

Adjoint Member: James Johnson from Wright Patterson Air Force Base

Chairman: Thomas G. Butler

After six months of investigation, the Ad Hoc Group reported to Headquarters that there was no digital program in existence which had broad, uniform capabilities

in the three interdependent disciplines of analytical mechanics, numerical methods, and computer programming. The Group did observe that there was considerable capability dispersed throughout the aerospace industry which had not been collected into a single program. They found also that there was a tendency toward proprietary secrecy which inhibited exchange of information. Communication was further hindered by the lack of compatibility between any two companies' structural analysis programs. The Ad Hoc Group recommended that NASA sponsor an entirely new program aimed at bringing together all the best in the state-of-the-arts. NASA is indebted to many people, but especially to these individuals in the several companies for assisting the Ad Hoc Group during its investigations:

M. John Turner of Boeing Seattle

W. J. Crichlaw of Lockheed Burbank

Paul H. Denke of Douglas Long Beach

Robert J. Melosh of Philco Western Development Labs

Richard H. Gallagher of Bell of Buffalo

P. L. Marshall of North American Columbus

Stanley Kaufman of Martin Baltimore

Headquarters endorsed the recommendations of the Ad Hoc Group and commissioned them to draw up a set of specifications. Fortunately, the papers from the first Wright Field Conference, "On Matrix Methods in Structural Analysis", were at hand to consult. The objectives of the specifications are now being achieved:

Combine the best of the state-of-the-arts in 3 disciplines.

Incorporate both the Force and the Displacement approaches of finite elements.

Organize to be General Purpose.

Embody large three dimensional structural capability.

Establish computer independence.

Provide for modification without cascading effects.

Build in the maximum of user convenience.

Document all aspects to gain maximum visibility.

The contract to implement the NASTRAN specifications was awarded to Computer Sciences Corporation (CSC) with MacNeal Schwendler, Martin Baltimore, and later

Bell Aerosystems Company as sub-contractors. The team that CSC assembled was one that identified strongly with the project. The designers were Dr. Richard H. MacNeal, Dr. Paul R. Peabody, Prof. C. W. McCormick, Mr. Stanley Kaufman, Mr. Thomas L. Clark, and Mr. David B. Hall. Except for Dr. Peabody, all of the designers were also involved with the implementation. The team of computer programmers was led by Mr. Keith H. Redner. The other principal members of the implementation team were Richard S. Pyle, Carl Hennrich, Steven E. Wall, Frank J. Douglas, Howard Dielmann, and David Herting. The quality of the NASTRAN program and its documentation is testimony to the purposefulness with which this implementation team applied themselves. We in NASA extend our sincere gratitude to all these men. This team has often exceeded the state-of-the-arts guidelines that were set down. A few examples are: the Segment File Allocator, the General Input/Output Module, matrix decomposition with active columns, the inclusion of scalar nonlinearities in control system dynamics, the generality in the plot module, and the development of the self-contained "inverse power with shifts" module for eigenvalue extraction. The overall design of the program has set a new standard for general purpose programs of any discipline. The framework used in NASTRAN can be disassociated from elastic structures and be applied to other disciplines, because there are no semantic implications in the executive operations. The program abounds in service code, that threads through every step of the problem physics, providing convenience to the analyst.

The NASTRAN Project personnel at Goddard Space Flight Center also strongly identified with the program. They are: Richard D. McConnell, William R. Case, James B. Mason, William L. Cook, and Edward F. Puccinelli. Their unflagging efforts in debugging and editing have been indispensable.

Many policy decisions had to be made as to the content of NASTRAN. The total framework of the program was considered to be the most important; i.e., commensurate large problem capability from basic statics for general conditions to advanced dynamics; with an executive system that could manage problems unbounded by core; was compact in its space requirements; could restart problems; could operate efficiently over different computers; and still be maintainable. This is

a costly venture and one could not expect the program to be complete in all its details. We wanted it to be complete in its principals and essentials. Consequently, some aspects were of necessity postponed.

It was decided that only basic finite elements would be included to deal with one and two dimensional elastic relationships such as beams, plates and axisymmetric shells. Economy of running time for the state-of-the-arts elements was a determining factor in their selection. More sophisticated and new elements have appeared in the state-of-the-art since the inception of this program, which can easily be incorporated.

An interesting type of decision erupted as a result of third generation computers. The relative costs of input-output and sorting operations had to be compared against regeneration whenever calculated data was used subsequently. In the majority of instances, the decision was made in favor of regeneration because the quantity of input data (usually from single precision engineering sources) was small; the compute time for regeneration was usually less than the associated input-output time; the quantity of generated data was usually orders of magnitude greater (and in double precision) compared to the engineering source data, and the read and write operations and subsequent logical operations for sorting and merging were very time-consuming.

In the matter of eigenvalues, the range of demands is such that no single current routine will satisfy all requirements. Two principal types of routines were decided upon; those that transform the whole matrix for simultaneous determination of roots and those that perform separate operations on the original matrix for each root extracted. The former is more efficient for a large number of roots, while the latter type is more efficient for a few roots of large matrices. Solution of complex matrices is performed only by the root tracking schemes and not by transformation methods, because existing transformation techniques for complex matrices are too expensive.

There was advice from many sources that the time was not right for a large general purpose program, because the flux in hardware and software was too great. The fear was that the endeavor would result in a program which would be obsolete

before it was finished. The rapidly changing computer technology had the opposite effect on the NASTRAN Project. The situation forced men to think more generally so that viability could be preserved in spanning the gap between second and third generation machines and anticipating the impact of fourth generation computers. The decision was made to write a single program in FORTRAN IV with some exceptional areas to be written in assembly language. FORTRAN IV version 13 in its various forms on different computers seems to have stabilized out as a language that will have currency for a reasonable time in the future. More than 99% of the program is written in this FORTRAN. The program is modular so that updating is a matter of revamping material within a module without modifying its external appearances.

The task of writing a program that performed efficiently on different manufacturers' computers forced many decisions. It was very tempting to capitulate to the disparities amongst the computers and resign ourselves to the rationale that for each computer, there would be a different version of NASTRAN. Fortunately, the decision was made to create just one version of NASTRAN. In some respects, this decision might have been viewed by some as having a corroding effect on the program. The conception derives from the condition that code was not written simultaneously for all computers. It was written first for the "development computer - the IBM Direct Coupled 7094/7040"; subsequently, it was tried on the others. Failures of programming designs on subsequent machines decreased the set of admissable procedures that remained in the intersection of the languages of all required computers. Fortunately, the remaining subset is sufficiently versatile, that it was possible to achieve commonality without becoming primitive. To date, the computers for which NASTRAN has been adopted are: IBM 7094/7044, UNIVAC 1108, CDC6600, and IBM 360. One serious lack of common intersection was in core management. NASTRAN is so large that logic demands that it be designed in multiple links horizontally and in many levels of overlay vertically with the additional ability to manage core dynamically. When it was found that Control Data did not provide for this type of core management, the decision was made to write a loader to replace that which Control Data offered with their SCOPE 3 operating system. The basic set of NASTRAN functional modules

is thus preserved as a single unit for all computers. Problems in trouble-shooting and maintenance are thus confined to a single source. This decision affected the compatibility with time sharing of the IBM TSS type. An entirely different version of NASTRAN would be required to adapt the program to the TSS system. Well-founded doubts also persist about its eventual execution efficiency. Consequently, NASTRAN is not designed to operate under a time sharing system.

Originally, a number of conveniences were going to be built into NASTRAN. As design evolved, the number of convenience nominees expanded. It was finally decided that conveniences having to do with the preparation of input data in NASTRAN format were properly external to the program. All such routines could be classified as NASTRAN auxiliary programs. Examples of this classification are: the automatic generation of grid points and elastic elements for an analytically known geometry, the computation of equation ordering for the minimization of the bandwidth of non zero elements in the stiffness matrix, or the formatting of experimental data as input to the General Element. Hundreds of such programs will probably be written and their lives will probably be short. For these reasons, they are considered external auxiliaries. Circulation of announcements about such programs will be made to users of the program, so they can be obtained easily.

Hopefully, the framework that has been built has also been sufficiently well fleshed out that it will serve a sizable portion of the large problems in the structural analysis community currently. It is intended that new capability be added or outdated capability be replaced by augmenting or replacing modules. An entire chapter in the Programmer's Manual has been devoted to the topic of Modifications and Additions in anticipation of serving this particular activity. The traffic in the development of new and increasingly versatile elastic modules is expected to be the most active. As analysts increase their use of the program, their detailed needs will become better defined with the consequent result that the traffic in "convenience code" will also be expected to increase. It is our strong desire that all such new features be called to the attention of the NASTRAN Project so that these ideas and routines can be disseminated to a broad audience.

Thomas G. Butler
NASTRAN Project Manager
Goddard Space Flight Center
Greenbelt, Maryland

PREFACE TO THE NASTRAN PROGRAMMER'S MANUAL

The Programmer's Manual is one of three manuals that constitute the documentation for NASTRAN, the other two being the Theoretical Manual and the User's Manual. The Programmer's Manual is divided into six major sections: section 1, NASTRAN Programming Fundamentals; section 2, Data Block and Table Descriptions; section 3, Subroutine Descriptions; section 4, Module Functional Descriptions; section 5, NASTRAN - Operating System Interfaces; and section 6, Modifications and Additions to NASTRAN.

Section 1 is a general overview of the program, and as such it should be read as background material for all sections which follow.

Section 2 contains descriptions of the data blocks, which are the principal means of data communication between the program's functional modules (a module is defined to be a group of subroutines which perform a specific function) and the NASTRAN Executive System. Two indexes for the data block descriptions, one sorted alphabetically on data block names and the other sorted alphabetically on the names of the modules from which the data blocks are output, are given in sections 2.2.1 and 2.2.2 respectively. Section 2 also contains a) descriptions of tables, both core and noncore resident, maintained by the NASTRAN Executive System and b) descriptions of miscellaneous tables which are accessed by a class of modules. Alphabetical indexes for these tables are given at the beginning of sections 2.4 and 2.5 respectively.

Sections 3 and 4 contain descriptions of the (utility or general purpose) subroutines and modules of NASTRAN respectively. The reader is directed to the alphabetical indexes, sorted on entry point names, in sections 3.2 and 4.1.3 respectively for these sections. An index to the Module Functional Descriptions, sorted alphabetically on module names, is given in section 4.1.2. The reader is urged to read the introductory material to sections 3 and 4 before using these sections.

Section 5 treats computer and operating system dependent matters such as operating system control cards and generation of the absolute (executable) NASTRAN system.

Section 6 describes the means by which modifications and additions to NASTRAN are implemented.

The learning of any new system, whether it be an operating system or a large applications system like NASTRAN, is made more difficult than it ought to be because of the use by the designers

of the system of new mnemonics, acronyms, phrases and "buzz" words. In order to aid the reader in learning such commonly used NASTRAN terms, a single source reference, section 7, the NASTRAN Dictionary, of the User's Manual is provided. The programmer is advised to secure a copy of at least this section of the User's Manual for his day-to-day reference.

TABLE OF CONTENTS

Section		Page No.
1.	NASTRAN PROGRAMMING FUNDAMENTALS	
1.1	PROGRAM OVERVIEW	1.1-1
1.1.1	Objectives	1.1-1
1.1.2	Program Organization	1.1-3
1.2	NASTRAN EXECUTIVE SYSTEM	1.2-1
1.2.1	Introduction	1.2-1
1.2.2	Executive Operations During the Preface	1.2-4
1.2.3	Executive Operations During Problem Solution	1.2-9
1.3	WORD SIZE AND COMPUTER HARDWARE CONSIDERATIONS	1.3-1
1.3.1	Introduction	1.3-1
1.3.2	Alphanumeric Data	1.3-2
1.3.3	Word Packing	1.3-2
1.4	SYSTEM BLOCK DATA SUBPROGRAM (SEMDBD)	1.4-1
1.5	THE OPEN CORE CONCEPT	1.5-1
1.5.1	Introduction	1.5-1
1.5.2	Definition of Open Core	1.5-1
1.5.3	Example of an Application of Open Core	1.5-1
1.6	NASTRAN INPUT/OUTPUT	1.6-1
1.6.1	Introduction	1.6-1
1.6.2	Use of the Operating System Input File	1.6-1
1.6.3	Use of the Operating System Output File	1.6-2
1.6.4	GINØ	1.6-3
1.7	NASTRAN MATRIX ROUTINES	1.7-1
1.7.1	Introduction	1.7-1
1.7.2	Matrix Packing and Unpacking	1.7-1
1.7.3	The Nested Vector Set Concept Used to Represent Components of Displacement	1.7-2
1.8	GENERATION OF MATRICES	1.8-1
1.8.1	The ECPT Data Block	1.8-1

Section		Page No.
	1.8.2 Structural Elements	1.8-2
1.9	TERMINATION PHILOSOPHY AND DIAGNOSTIC MESSAGES	1.9-1
1.10	RESTARTS IN NASTRAN	1.10-1
2.	DATA BLOCK AND TABLE DESCRIPTIONS	
2.1	INTRODUCTION	2.1-1
2.2	DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES	2.2-1
	2.2.1 Index for Data Block Descriptions Sorted on Data Block Names	2.2-3
	2.2.2 Index for Data Block Descriptions Sorted Alphabetically by Module	2.2-11
2.3	DATA BLOCK DESCRIPTIONS	2.3-1
	2.3.1 Data Blocks Output From Module IFP1	2.3-1
	2.3.2 Data Blocks Output From Module IFP	2.3-5
	2.3.3 Data Blocks Output From Module GP1	2.3-31
	2.3.4 Data Blocks Output From Module GP2	2.3-36
	2.3.5 Data Blocks Output From Module PLTSET	2.3-37
	2.3.6 Data Blocks Output From Module PLØT	2.3-40
	2.3.7 Data Blocks Output From Module GP3	2.3-41
	2.3.8 Data Blocks Output From Module TA1	2.3-45
	2.3.9 Data Blocks Output From Module SMA1	2.3-56
	2.3.10 Data Blocks Output From Module SMA2	2.3-58
	2.3.11 Data Blocks Output From Module GPWG	2.3-59
	2.3.12 Data Blocks Output From Module SMA3	2.3-60
	2.3.13 Data Blocks Output From Module GP4	2.3-61
	2.3.14 Data Blocks Output From Module GPSP	2.3-63
	2.3.15 Data Blocks Output From Module MCE1	2.3-64
	2.3.16 Data Blocks Output From Module MCE2	2.3-65
	2.3.17 Data Blocks Output From Module SCE1	2.3-67
	2.3.18 Data Blocks Output From Module SMP1	2.3-70
	2.3.19 Data Blocks Output From Module RBMG1	2.3-73
	2.3.20 Data Blocks Output From Module RBMG2	2.3-75
	2.3.21 Data Blocks Output From Module RBMG3	2.3-77
	2.3.22 Data Blocks Output From Module RBMG4	2.3-78

Section		Page No.
2.3.23	Data Blocks Output From Module SSG1	2.3-79
2.3.24	Data Blocks Output From Module SSG2	2.3-80
2.3.25	Data Blocks Output From Module SSG3	2.3-81
2.3.26	Data Blocks Output From Module SSG4	2.3-83
2.3.27	Data Blocks Output From Module SDR1	2.3-84
2.3.28	Data Blocks Output From Module SDR2	2.3-88
2.3.29	Data Blocks Output From Module DPD	2.3-114
2.3.30	Data Blocks Output From Module READ	2.3-125
2.3.31	Data Blocks Output From Module DSMG1	2.3-128
2.3.32	Data Blocks Output From Module SMP2	2.3-129
2.3.33	Data Blocks Output From Module DSMG2	2.3-130
2.3.34	Data Blocks Output From Module PLA1	2.3-132
2.3.35	Data Blocks Output From Module ADD	2.3-137
2.3.36	Data Blocks Output From Module PLA2	2.3-138
2.3.37	Data Blocks Output From Module PLA3	2.3-139
2.3.38	Data Blocks Output From Module PLA4	2.3-140
2.3.39	Data Blocks Output From Module CASE	2.3-141
2.3.40	Data Blocks Output From Module MTRXIN	2.3-142
2.3.41	Data Blocks Output From Module GKAD	2.3-143
2.3.42	Data Blocks Output From Module CEAD	2.3-146
2.3.43	Data Blocks Output From Module VDR	2.3-149
2.3.44	Data Blocks Output From Module FRRD	2.3-158
2.3.45	Data Blocks Output From Module SDR3	2.3-160
2.3.46	Data Blocks Output From Module XYTRAN	2.3-175
2.3.47	Data Blocks Output From Module RANDØM	2.3-179
2.3.48	Data Blocks Output From Module TRD	2.3-181
2.3.49	Data Blocks Output From Module GKAM	2.3-183
2.3.50	Data Blocks Output From Module DDR1	2.3-184
2.3.51	Element Stress Output Data Description	2.3-185
2.3.52	Element Force Output Data Description	2.3-189

Section		Page No.
	2.3.53 Data Blocks Output From Module DDR2	2.3-192
2.4	EXECUTIVE TABLE DESCRIPTIONS	2.4-1
	2.4.1 Executive Tables Which are Permanently Core Resident	2.4-2
	2.4.2 Executive Tables Not Permanently Core Resident	2.4-15
2.5	MISCELLANEOUS TABLE DESCRIPTIONS	2.5-1
	2.5.1 Miscellaneous Tables Which are Permanently Core Resident	2.5-2
	2.5.2 Miscellaneous Tables Not Permanently Core Resident	2.5-6
3.	SUBROUTINE DESCRIPTIONS	
3.1	INTRODUCTION	3.1-1
3.2	ALPHABETICAL INDEX OF ENTRY POINTS FOR SUBROUTINE DESCRIPTIONS	3.2-1
3.3	EXECUTIVE SUBROUTINE DESCRIPTIONS	3.3-1
	3.3.1 XSEM1 (Executive Sequence Monitor, Preface)	3.3-1
	3.3.2 BTSTRP (Bootstrap Generator)	3.3-2
	3.3.3 SEMINT (Sequence Monitor Initialization)	3.3-3
	3.3.4 GNFIAT (Generate FIAT)	3.3-5
	3.3.5 ENDSYS (End-of-Link)	3.3-6
	3.3.6 SEARCH (Search, Load, and Execute Link)	3.3-8
	3.3.7 XSEMi (Link i Main Program, i = 2,3,...)	3.3-9
	3.3.8 XSEMXX (Sequence Monitor - Deck Generator)	3.3-11
	3.3.9 GNFIST (Generate FIST)	3.3-12
	3.3.10 XEØT (End-of-Tape)	3.3-14
	3.3.11 SSWTCH (Sense Switches)	3.3-15
	3.3.12 CØNMSG (Console Message Writer)	3.3-16
	3.3.13 TTLPRG (Title Page Writer)	3.3-17
3.4	UTILITY SUBROUTINE DESCRIPTIONS	3.4-1
	3.4.1 MAPFNS (Machine Word Functions)	3.4-1
	3.4.2 ØPEN (Initiate Activity on a File)	3.4-3
	3.4.3 WRITE (Write Data in a Logical Record)	3.4-4
	3.4.4 CLØSE (Terminate Activity on a File)	3.4-5
	3.4.5 READ (Read Data From a Logical Record)	3.4-6

Section		Page No.
3.4.6	FWDREC (Forward Space One Logical Record)	3.4-8
3.4.7	BCKREC (Backspace One Logical Record)	3.4-9
3.4.8	REWIND (Position File to the Load Point)	3.4-10
3.4.9	EØF (Write an End-of-File)	3.4-11
3.4.10	SKPFIL (Skip Files Forward or Backward)	3.4-12
3.4.11	XGINØ (GINØ Utility Routine)	3.4-13
3.4.12	GINØ (General Input/Output Routine)	3.4-15
3.4.13	ØPNCØR (Transmit Logical Records To/From Core Storage)	3.4-20
3.4.14	GØPEN (Short Form for Subroutine ØPEN With Header Record Processing)	3.4-22
3.4.15	FREAD (Short Form for Subroutine READ)	3.4-23
3.4.16	WRTTRL (Write Trailer)	3.4-24
3.4.17	FNAME (File Name)	3.4-25
3.4.18	CLSTAB (Close a GINØ File and Write a Nonzero Trailer)	3.4-26
3.4.19	XRCARD (Executive Free-Field Card Data Conversion Routine)	3.4-27
3.4.20	RCARD (Fixed Field Card Data Conversion Routine)	3.4-32
3.4.21	TAPBIT (Tape Bit Test)	3.4-35
3.4.22	PEXIT (Problem Exit)	3.4-36
3.4.23	TMTØGØ (Time-To-Go)	3.4-37
3.4.24	PAGE (Page Heading)	3.4-38
3.4.25	MESAGE (Message)	3.4-39
3.4.26	MSGWRT (Message Writer)	3.4-40
3.4.27	USRMSG (User Message Writer)	3.4-41
3.4.28	MATDUM (Matrix Dump (Print) Routine)	3.4-42
3.4.29	TABPRT (Table Printer)	3.4-43
3.4.30	PRELØC (Position Data Block to Requested Record)	3.4-44
3.4.31	SØRT (Sort a Table)	3.4-46
3.4.32	GMMATD (General Matrix Multiply and Transpose - Double Precision)	3.4-49
3.4.33	GMMATS (General Matrix Multiply and Transpose - Single Precision)	3.4-52
3.4.34	INVERD (Double Precision In Core Inverse Routine)	3.4-53

Section		Page No.
3.4.35	INVERS (Single Precision In Core Inverse Routine)	3.4-54
3.4.36	PREMAT (Material Property Utility)	3.4-55
3.4.37	PRETRD (Utility for Modules Which Use the CSTM Data Block - Double Precision Version)	3.4-64
3.4.38	PRETRS (Utility for Modules Which Use the CSTM Data Block - Single Precision Version)	3.4-66
3.4.39	PRETAB (Table Look-Up)	3.4-67
3.4.40	AXIS (Draw an Axis on a Plot)	3.4-70
3.4.41	AXISi (Axis Routine for Plotter i)	3.4-72
3.4.42	SKPFRM (Skip a Variable Number of Frames)	3.4-73
3.4.43	SELCAM (To Initiate a New Plot)	3.4-74
3.4.44	IDPLØT (Generate an "ID" Plot)	3.4-75
3.4.45	INTGPX (Search a List of Integers)	3.4-76
3.4.46	INTLST (Interpret a List of Integers)	3.4-77
3.4.47	LINE (Draw a Line on a Plotter)	3.4-78
3.4.48	LINEi (Draw a Line on Plotter i)	3.4-79
3.4.49	PRINT (Print a Title on a Plotter)	3.4-81
3.4.50	RDMØDX (Read a File Containing XRCARD Translations)	3.4-83
3.4.51	SGINØ (GINØ for Unformatted Tapes)	3.4-85
3.4.52	STPLØT (To Initiate a New Plot or Terminate the Current Plot)	3.4-87
3.4.53	SYMBØL (Type a Symbol on a Plotter)	3.4-88
3.4.54	TIPE (Type a Line of Characters on a Plotter)	3.4-90
3.4.55	TYPEi (Type a Line of Characters on Plotter i)	3.4-92
3.4.56	TYPFLT (Type a Floating Point Number on a Plotter)	3.4-94
3.4.57	TYPINT (Type an Integer Number on a Plotter)	3.4-96
3.4.58	WPLT1 (Write a Plotter Command for Plotter 1)	3.4-98
3.4.59	WPLT2 (Write a Plotter Command for Plotters 2 and 8)	3.4-100
3.4.60	WPLT3 (Write a Plotter Command for Plotter 3)	3.4-102
3.4.61	GINØIØ (GINØ Input/Output Routine)	3.4-103
3.4.62	EJECT (Automatic Page Eject)	3.4-105
3.4.63	PLAMAT (Material Property Utility for Two-Dimensional Elements in Piecewise Linear Analysis)	3.4-106

Section			Page No.
	3.4.64	WPLT4 (Write a Plotter Command for Plotters 4 through 7).....	3.4-108
	3.4.65	WPLT9 (Write a Plotter Command for Plotter 9).....	3.4-110
	3.4.66	WPLT10 (Write a Plotter Command for the NASTRAN General Purpose Plotter).....	3.4-111
	3.4.67	PLTSET (Plotting Parameter Initialization).....	3.4-113
	3.4.68	DRWCHR (To Draw a Line of Characters).....	3.4-115
	3.4.69	FNDPLT (Determine the Internal Plotter and Model Indices).....	3.4-117
3.5		MATRIX SUBROUTINE DESCRIPTIONS.....	3.5-1
	3.5.1	BLDPK (Build a Packed Column of a Matrix).....	3.5-1
	3.5.2	PACK (Pack a Column of a Matrix).....	3.5-5
	3.5.3	INTPK (Interpret a Packed Column of a Matrix).....	3.5-7
	3.5.4	UNPACK (Unpack a Packed Column of a Matrix).....	3.5-10
	3.5.5	CALCV (Compute a Partitioning Vector).....	3.5-12
	3.5.6	PARTN - MERGE (Partition a Matrix - Merge Matrices Together).....	3.5-13
	3.5.7	SSG2A (Driver for PARTN).....	3.5-16
	3.5.8	SDR1B (Driver for MERGE).....	3.5-17
	3.5.9	UPART (Symmetric Partition Driver).....	3.5-18
	3.5.10	ADD (Matrix Addition Routine).....	3.5-19
	3.5.11	SSG2C (Driver for ADD).....	3.5-20
	3.5.12	MPYAD (Matrix Multiplication Routine).....	3.5-22
	3.5.13	SSG2B (Driver for MPYAD).....	3.5-29
	3.5.14	SDCØMP (Symmetric Decomposition).....	3.5-30
	3.5.15	DECØMP (Unsymmetric Matrix Decomposition).....	3.5-44
	3.5.16	CDCØMP (Complex Matrix Decomposition).....	3.5-62
	3.5.17	FBS (Forward - Backward Substitution).....	3.5-64
	3.5.18	SSG3A (Driver for FBS).....	3.5-66
	3.5.19	GFBS (General Forward - Backward Substitution).....	3.5-67
	3.5.20	SØLVER (Simultaneous Equation Solution Routine).....	3.5-69
	3.5.21	DMPY (Multiply a Diagonal Matrix by an Arbitrary Matrix).....	3.5-71
	3.5.22	ELIM (Perform a Matrix Reduction).....	3.5-73
	3.5.23	FACTØR (Decompose a Matrix Into Triangular Factors).....	3.5-74

Section		Page No.
	3.5.24 TRANP1 (Driver for TRNSP)	3.5-75
	3.5.25 TRNSP (Matrix Transpose)	3.5-76
4.	MODULE FUNCTIONAL DESCRIPTIONS	
4.1	GENERAL COMMENTS AND INDEXES	4.1-1
	4.1.1 Use of Module Functional Descriptions	4.1-2
	4.1.2 Alphabetical Index of Module Functional Descriptions	4.1-7
	4.1.3 Alphabetical Index of Entry Points in Module Functional Descriptions	4.1-8
4.2	EXECUTIVE PREFACE MODULE XCSA (EXECUTIVE CONTROL SECTION ANALYSIS)	4.2-1
4.3	EXECUTIVE PREFACE MODULE IFP1 (INPUT FILE PROCESSOR, PART 1)	4.3-1
4.4	EXECUTIVE PREFACE MODULE XSØRT (EXECUTIVE BULK DATA CARD SORT)	4.4-1
4.5	EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)	4.5-1
4.6	EXECUTIVE PREFACE MODULE IFP3 (INPUT FILE PROCESSOR 3)	4.6-1
4.7	EXECUTIVE PREFACE MODULE XGPI (EXECUTIVE GENERAL PROBLEM INITIALIZATION)	4.7-1
4.8	EXECUTIVE PREFACE MODULE UMFEDIT (USER MASTER FILE EDITOR)	4.8-1
4.9	EXECUTIVE MODULE XSFA (EXECUTIVE SEGMENT FILE ALLOCATOR)	4.9-1
4.10	EXECUTIVE DMAP MODULE CHKPNT (CHECKPOINT)	4.10-1
4.11	EXECUTIVE DMAP INSTRUCTION REPT (REPEAT A GROUP OF DMAP INSTRUCTIONS)	4.11-1
4.12	EXECUTIVE DMAP INSTRUCTION JUMP (UNCONDITIONAL DMAP TRANSFER)	4.12-1
4.13	EXECUTIVE DMAP INSTRUCTION CØND (CONDITIONAL TRANSFER)	4.13-1
4.14	EXECUTIVE DMAP INSTRUCTION EXIT (TERMINATE DMAP PROGRAM)	4.14-1
4.15	EXECUTIVE DMAP MODULE SAVE (SAVE VARIABLE PARAMETER VALUES)	4.15-1
4.16	EXECUTIVE DMAP MODULE PURGE (EXPLICIT DATA BLOCK PURGE)	4.16-1
4.17	EXECUTIVE DMAP MODULE EQUIV (DATA BLOCK NAME EQUIVALENCE)	4.17-1
4.18	EXECUTIVE DMAP INSTRUCTION END (END OF DMAP PROGRAM)	4.18-1
4.19	EXECUTIVE DMAP MODULE PARAM (PARAMETER PROCESSOR)	4.19-1
4.20	EXECUTIVE DMAP MODULE SETVAL (SET VALUES)	4.20-1
4.21	FUNCTIONAL MODULE GP1 (GEOMETRY PROCESSOR - PHASE 1)	4.21-1
4.22	FUNCTIONAL MODULE GP2 (GEOMETRY PROCESSOR - PHASE 2)	4.22-1
4.23	FUNCTIONAL MODULE PLTSET (PLOT SET DEFINITION PROCESSOR)	4.23-1
4.24	FUNCTIONAL MODULE PLØT (STRUCTURAL PLOTTER)	4.24-1

Section		Page No.
4.25	FUNCTIONAL MODULE GP3 (GEOMETRY PROCESSOR - PHASE 3).....	4.25-1
4.26	FUNCTIONAL MODULE TA1 (TABLE ASSEMBLER).....	4.26-1
4.27	FUNCTIONAL MODULE SMA1 (STRUCTURAL MATRIX ASSEMBLER - PHASE 1).....	4.27-1
4.28	FUNCTIONAL MODULE SMA2 (STRUCTURAL MATRIX ASSEMBLER - PHASE 2).....	4.28-1
4.29	FUNCTIONAL MODULE GPWG (GRID POINT WEIGHT GENERATOR).....	4.29-1
4.30	FUNCTIONAL MODULE SMA3 (STRUCTURAL MATRIX ASSEMBLER - PHASE 3).....	4.30-1
4.31	FUNCTIONAL MODULE GP4 (GEOMETRY PROCESSOR - PHASE 4).....	4.31-1
4.32	FUNCTIONAL MODULE GPSP (GRID POINT SINGULARITY PROCESSOR).....	4.32-1
4.33	FUNCTIONAL MODULE MCE1 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 1).....	4.33-1
4.34	FUNCTIONAL MODULE MCE2 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 2).....	4.34-1
4.35	FUNCTIONAL MODULE SCE1 (SINGLE-POINT CONSTRAINT ELIMINATOR).....	4.35-1
4.36	FUNCTIONAL MODULE SMP1 (STRUCTURAL MATRIX PARTITIONER - PHASE 1).....	4.36-1
4.37	FUNCTIONAL MODULE RBMG1 (RIGID BODY MATRIX GENERATOR - PHASE 1).....	4.37-1
4.38	FUNCTIONAL MODULE RBMG2 (RIGID BODY MATRIX GENERATOR - PHASE 2).....	4.38-1
4.39	FUNCTIONAL MODULE RBMG3 (RIGID BODY MATRIX GENERATOR - PHASE 3).....	4.39-1
4.40	FUNCTIONAL MODULE RBMG4 (RIGID BODY MATRIX GENERATOR - PHASE 4).....	4.40-1
4.41	FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1).....	4.41-1
4.42	FUNCTIONAL MODULE SSG2 (STATIC SOLUTION GENERATOR - PHASE 2).....	4.42-1
4.43	FUNCTIONAL MODULE SSG3 (STATIC SOLUTION GENERATOR - PHASE 3).....	4.43-1
4.44	FUNCTIONAL MODULE SSG4 (STATIC SOLUTION GENERATOR - PHASE 4).....	4.44-1
4.45	FUNCTIONAL MODULE SDR1 (STRESS DATA RECOVERY - PHASE 1).....	4.45-1
4.46	FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2).....	4.46-1
4.47	FUNCTIONAL MODULE DPD (DYNAMICS POOL DISTRIBUTOR).....	4.47-1
4.48	FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT).....	4.48-1
4.49	FUNCTIONAL MODULE DSMG1 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR - PHASE 1).	4.49-1
4.50	FUNCTIONAL MODULE SMP2 (STRUCTURAL MATRIX PARTITIONER - PHASE 2).....	4.50-1
4.51	FUNCTIONAL MODULE DSMG2 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR - PHASE 2).	4.51-1
4.52	FUNCTIONAL MODULE PLA1 (PIECEWISE LINEAR ANALYSIS - PHASE 1).....	4.52-1
4.53	FUNCTIONAL MODULE PLA2 (PIECEWISE LINEAR ANALYSIS - PHASE 2).....	4.53-1
4.54	FUNCTIONAL MODULE PLA3 (PIECEWISE LINEAR ANALYSIS - PHASE 3).....	4.54-1

Section		Page No.
4.55	FUNCTIONAL MODULE PLA4 (PIECEWISE LINEAR ANALYSIS - PHASE 4).....	4.55-1
4.56	FUNCTIONAL MODULE CASE (SIMPLIFY CASE CONTROL).....	4.56-1
4.57	FUNCTIONAL MODULE MTRXIN (MATRIX INPUT).....	4.57-1
4.58	FUNCTIONAL MODULE GKAD (GENERAL K ASSEMBLER DIRECT).....	4.58-1
4.59	FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT).....	4.59-1
4.60	FUNCTIONAL MODULE VDR (VECTOR DATA RECOVERY).....	4.60-1
4.61	FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH).....	4.61-1
4.62	FUNCTIONAL MODULE SDR3 (STRESS DATA RECOVERY - PHASE 3 - SØRT1 to SØRT2..... PROCESSOR).....	4.62-1
4.63	FUNCTIONAL MODULE XYTRAN (XY - OUTPUT DATA TRANSLATOR).....	4.63-1
4.64	FUNCTIONAL MODULE RANDØM (RANDOM ANALYSIS MODULE).....	4.64-1
4.65	FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT).....	4.65-1
4.66	FUNCTIONAL MODULE GKAM (GENERAL K ASSEMBLER MODAL).....	4.66-1
4.67	FUNCTIONAL MODULE DDR1 (DYNAMIC DATA RECOVERY - PART 1).....	4.67-1
4.68	FUNCTIONAL MODULE DDR2 (DYNAMIC DATA RECOVERY - PART 2).....	4.68-1
4.69	OUTPUT MODULE XYPLØT (X-Y DATA PLOTTER).....	4.69-1
4.70	OUTPUT MODULE ØFP (OUTPUT FILE PROCESSOR).....	4.70-1
4.71	OUTPUT MODULE MATPRN (GENERAL MATRIX PRINTER).....	4.71-1
4.72	OUTPUT MODULE MATGPR (DISPLACEMENT METHOD MATRIX PRINTER).....	4.72-1
4.73	OUTPUT MODULE MATPRT (MATRIX PRINTER).....	4.73-1
4.74	OUTPUT MODULE SEEMAT (PICTORIAL MATRIX PRINTER).....	4.74-1
4.75	OUTPUT MODULE TABPT (TABLE PRINTER).....	4.75-1
4.76	OUTPUT MODULE PRTMSG (MESSAGE WRITER).....	4.76-1
4.77	OUTPUT MODULE PRTPARM (PARAMETER AND DMAP MESSAGE PRINTER).....	4.77-1
4.78	MATRIX MODULE ADD (ADD TWO MATRICES).....	4.78-1
4.79	MATRIX MODULE MPYAD (MULTIPLY ADD).....	4.79-1
4.80	MATRIX MODULE SØLVE (SOLVES THE MATRIX EQUATION [A][X] = [B]).....	4.80-1
4.81	MATRIX MODULE DECØMP (MATRIX DECOMPOSITION).....	4.81-1
4.82	MATRIX MODULE FBS (FORWARD - BACKWARD SUBSTITUTION).....	4.82-1
4.83	MATRIX MODULE PARTN (PARTITION A MATRIX).....	4.83-1
4.84	MATRIX MODULE MERGE (MERGE MATRICES TOGETHER).....	4.84-1

Section		Page No.
4.85	MATRIX MODULE TRNSP (TRANSPOSE A MATRIX)	4.85-1
4.86	MATRIX MODULE SMPYAD (STRING MULTIPLY ADD)	4.86-1
4.87	STRUCTURAL ELEMENT DESCRIPTIONS	4.87-1
4.87.1	The RØD, CØNRØD, and TUBE Elements	4.87-7
4.87.1.1	Input Data for the RØD, TUBE, CØNRØD Elements	4.87-7
4.87.1.2	Stiffness Matrix Calculation (Subroutines KRØD and KTUBE of Module SMA1)	4.87-8
4.87.1.3	Lumped Mass Matrix Calculation (Subroutines MRØD and MTUBE of Module SMA2)	4.87-9
4.87.1.4	Element Load Calculations (Subroutine EDTL of Module SSG1)	4.87-10
4.87.1.5	Element Stress Calculations (Subroutines SRØD1 and SRØD2 of Module SDR2)	4.87-10
4.87.1.6	Differential Stiffness Matrix Calculation (Subroutine DRØD of Module DSMG1)	4.87-12
4.87.1.7	Piecewise Linear Analysis Calculations (Subroutine PSRØD of Module PLA3 and Subroutine PKRØD of Module PLA4)	4.87-14
4.87.1.8	Coupled Mass Matrix Calculation (Subroutine MCRØD of Module SMA2)	4.87-16a
4.87.2	The BAR Element	4.87-17
4.87.2.1	Input Data for the BAR Element	4.87-17
4.87.2.2	Stiffness Matrix Calculation (Subroutine KBAR of Module SMA1)	4.87-18
4.87.2.3	Lumped Mass Matrix Calculation (Subroutine MBAR of Module SMA2)	4.87-25
4.87.2.4	Element Load Calculation (Subroutine BAR of Module SSG1)	4.87-26
4.87.2.5	Element Stress Calculations (Subroutines SBAR1 and SBAR2 of Module SDR2)	4.87-27
4.87.2.6	Differential Stiffness Matrix Calculation (Subroutine DBEAM of Module DSMG1)	4.87-29
4.87.2.7	Piecewise Linear Analysis Calculations (Subroutine PSBAR of Module PLA3 and Subroutine PKBAR of Module PLA4)	4.87-32
4.87.2.8	"Consistent" Mass Matrix Calculation (Subroutine MCBAR of Module SMA2)	4.87-36
4.87.3	The SHEAR Panel and TWIST Panel Elements	4.87-38

Section			Page No.
	4.87.3.1	Input Data for SHEAR and TWIST Panels	4.87-38
	4.87.3.2	Definition of Element Geometry	4.87-39
	4.87.3.3	Coefficient Generation	4.87-41
	4.87.3.4	Stiffness Matrix Formulation For a SHEAR Panel (Subroutine KPANEL of Module SMA1)	4.87-46
	4.87.3.5	TWIST Element Stiffness Matrix Generation (Subroutine KPANEL of Module SMA1)	4.87-47
	4.87.3.6	Mass Matrix Generation (Subroutine MASSTQ of Module SMA2)	4.87-48
	4.87.3.7	SHEAR Element Stress and Force Calculations (Subroutine SPANL1 and SPANL2 of Module SDR2)	4.87-50
	4.87.3.8	TWIST Element Stress and Force Calculations (Subroutines SPANL1 and SPANL2 of Module SDR2)	4.87-52
	4.87.3.9	SHEAR Panel Differential Stiffness Calculations (Subroutine DSHEAR of Module DSGM1)	4.87-54
4.87.4	TRMEM and QDMEM Elements		4.87-58
	4.87.4.1	Input Data for the TRMEM and QDMEM Elements	4.87-58
	4.87.4.2	Basic Equations for TRMEM	4.87-59
	4.87.4.3	Stiffness Matrix Calculation for TRMEM (Subroutine KTRMEM of Module SMA1)	4.87-61
	4.87.4.4	Mass Matrix Calculation for the TRMEM Element (Subroutine MASSTQ of Module SMA2)	4.87-62
	4.87.4.5	Element Load Calculations for the TRMEM Element (Subroutine TRIMEM of Module SSG1)	4.87-63
	4.87.4.6	Element Stress Calculations for the TRMEM Element (Subroutines STRME1 and STQME2 of Module SDR2)	4.87-63
	4.87.4.7	Differential Stiffness Matrix Calculations for the TRMEM Element (Subroutine DTRMEM of Module DSMG1)	4.87-67
	4.87.4.8	General Calculations for the QDMEM by the QDMEM Driver Routines (Subroutines KQDMEM of Module SMA1, SQDME1 of Module SDR2, DQDMEM of Module DSMG1)	4.87-67
	4.87.4.9	Stiffness Matrix Calculations for the QDMEM	4.87-70
	4.87.4.10	Element Stress Calculations for the QDMEM (Subroutine SQDME1 and STQME2 of Module SDR2)	4.87-70
	4.87.4.11	Mass Matrix Generation for the QDMEM Element (Subroutine MASSTQ of Module SMA2)	4.87-74
	4.87.4.12	Thermal Load Computation for the QDMEM	4.87-76
	4.87.4.13	Differential Stiffness Computations for the QDMEM (Subroutines DQDMEM and DTRMEM of Module DSGM1)	4.87-76

Section			Page No.
	4.87.4.14	Piecewise Linear Analysis Calculations (Subroutines PSTRM and PSQDM of Module PLA3 and Subroutines PKTRM PKTRM and PKQDM of Module PLA4).....	4.87-76a
4.87.5		The TRBSC, TRPLT and QDPLT Elements.....	4.87-78
	4.87.5.1	Input Data for the TRBSC and TRPLT Elements.....	4.87-78
	4.87.5.2	General Calculation for the TRBSC Element.....	4.87-79
	4.87.5.3	Stiffness Matrix Calculations for the TRBSC Element (Subroutine KTRBSC of Module SMA1).....	4.87-84
	4.87.5.4	Stress Calculations for the TRBSC Element.....	4.87-85
	4.87.5.5	Stiffness Matrix Calculations for the TRPLT Element (Subroutine KTRPLT of Module SMA1).....	4.87-87
	4.87.5.6	Structural Damping Matrices for the TRPLT Element.....	4.87-95
	4.87.5.7	Stress and Element Force Calculations for the TRPLT Element (Subroutines STRPL1 and SBSPL2 of Module SDR2)...	4.87-95
	4.87.5.8	Stiffness Matrix Calculations for the QDPLT Element (Subroutine KQDPLT of Module SMA1).....	4.87-97
	4.87.5.9	Stress and Element Force Calculations for the QDPLT Element (Subroutines SQDPL1 and SBSPL2 of Module SDR2)...	4.87-102
	4.87.5.10	Lumped Mass Matrix Generation for the TRBSC, TRPLT, and QDPLT Elements (Subroutine MASSTQ of Module SMA2).....	4.87-104
	4.87.5.11	Coupled Mass Matrix Calculations for the TRBSC Element (Subroutine MTRBSC of Module SMA2).....	4.87-104a
4.87.6		The TRIA1, TRIA2, QUAD1 and QUAD2 Elements.....	4.87-106
	4.87.6.1	Input Data for the TRIA1, TRIA2, QUAD1 and QUAD2.....	4.87-106
	4.87.6.2	Stiffness Matrix Calculations (Subroutine KTRIQD of Module SMA1).....	4.87-107
	4.87.6.3	Lumped Mass Matrix Generation (Subroutine MASSTQ of Module SMA2).....	4.87-108
	4.87.6.4	Thermal Load Calculations (Subroutine EDTL of Module SSG1).....	4.87-108
	4.87.6.5	Element Stress and Force Calculations (Subroutines STRQD1 and STRQD2 of Module SDR2).....	4.87-108
	4.87.6.6	Differential Stiffness Matrix Calculations.....	4.87-109
	4.87.6.7	Coupled Mass Matrix Calculations (Subroutine MTRIQD of Module SMA2).....	4.87-109
	4.87.6.8	Piecewise Linear Analysis Calculations (Subroutines PSTRI1, PSTRI2, PSQAD1, and PSQAD2 of Module PLA3 and PKTRI1, PKTRI2, PKQAD1 and PKQAD2 of Module PLA4).....	4.87-109a

Section			Page No.
4.87.7	The ELASi, MASSi and DAMPi Elements		4.87-110
	4.87.7.1	Input Data for the ELASi, MASSi and DAMPi Elements	4.87-110
	4.87.7.2	ELASi Stiffness Matrix Generation (Subroutine KELAS of Module SMA1)	4.87-110
	4.87.7.3	MASSi Mass Matrix Generation (Subroutine MASSD of Module SMA2)	4.87-111
	4.87.7.4	DAMPi Damping Matrix Generation (Subroutine MASSD of Module SMA2)	4.87-111
	4.87.7.5	ELASi Stress and Force Recovery (Subroutines SELAS1 and SELAS2 of Module SDR2)	4.87-111
4.87.8	Concentrated Mass Elements CØNM1, CØNM2		4.87-113
	4.87.8.1	ECPT Entries for the CØNM1 Mass Element	4.87-113
	4.87.8.2	Mass Matrix Calculations for the CØNM1 Element (Subroutine MCØNMX of Module SMA2)	4.87-113
	4.87.8.3	ECPT Entries for the CØNM2 Mass Element	4.87-114
	4.87.8.4	Mass Matrix Calculations for the CØNM2 Element (Subroutine MCØNMX of Module SMA2)	4.87-114
4.87.9	The CØNEAX Element		4.87-117
	4.87.9.1	Input Data for the CØNEAX Element	4.87-117
	4.87.9.2	Stiffness Matrix Calculations (Subroutine KCØNE of Module SMA1)	4.87-117
	4.87.9.3	Mass Matrix Computation (Subroutine MCØNE of Module SMA2)	4.87-118
	4.87.9.4	Element Load Calculations (Subroutine CØNE of Module SSG1)	4.87-118
	4.87.9.5	Element Stress Calculations (Subroutines SCØNE1, SCØNE2, SCØNE3 of Module SDR2)	4.87-123
	4.87.9.6	Differential Stiffness Matrix Calculations (Subroutine DCØNE of Module DSMG1)	4.87-127a
4.87.10	The TRIARG Element		4.87-128
	4.87.10.1	Input Data for the TRIARG Element	4.87-128
	4.87.10.2	General Geometric Calculations	4.87-129
	4.87.10.3	Integral Calculations	4.87-130
	4.87.10.4	Elastic Constants Matrix Calculations	4.87-132
	4.87.10.5	Stiffness Matrix Generation (Subroutine KTRIRG of Module SMA1)	4.87-133

Section			Page No.
	4.87.10.6	Mass Matrix Calculations (Subroutine MTRIRG of Module SMA2)	4.87-135
	4.87.10.7	Thermal Load Calculations (Subroutine TTRIRG of Module SSG1)	4.87-136
	4.87.10.8	Element Force and Stress Calculaticns (Subroutines STRIR1 and STRIR2 of Module SDR2)	4.87-136
4.87.11	The TRAPRG Element		4.87-139
	4.87.11.1	Input Data for the TRAPRG Element	4.87-139
	4.87.11.2	General Calculations	4.87-140
	4.87.11.3	Integral Calculations	4.87-142
	4.87.11.4	Elastic Constants Matrix Calculation	4.87-144
	4.87.11.5	Stiffness Matrix Generation (Subroutine KTRAPR of Module SMA1)	4.87-144
	4.87.11.6	Mass Matrix Calculation (Subroutine MTRAPR of Module SMA2)	4.87-146
	4.87.11.7	Thermal Load Calculations (Subroutine TTRAPR of Module SSG1)	4.87-147
	4.87.11.8	Element Force and Stress Calculations (Subroutines STRAP1 and STRAP2 of Module SDR2)	4.87-148
4.87.12	The TØRDRG Element		4.87-152
	4.87.12.1	Input Data for the TØRDRG Element	4.87-152
	4.87.12.2	General Calculations	4.87-153
	4.87.12.3	Integral Calculations	4.87-156
	4.87.12.4	Elastic Constants Matrix Calculations	4.87-160
	4.87.12.5	Stiffness Matrix Calculations (Subroutine KTØRDR of Module SMA1)	4.87-160
	4.87.12.6	Mass Matrix Calculations (Subroutine MTØRDR of Module SMA2)	4.87-165
	4.87.12.7	Thermal Load Calculations (Subroutine TTØRDR of Module SSG1)	4.87-166
	4.87.12.8	Element Force and Stress Calculations (Subroutines STØRD1 and STØRD2 of Module SDR2)	4.87-168
4.87.13	The VISC Element		4.87-175
	4.87.13.1	Input Data for the VISC Element	4.87-175
	4.87.13.2	Damping Matrix Calculations (Subroutine BVISC of Module SMA2)	4.87-175

Section		Page No.
4.87.14	Integral Calculations for the TRIARG, TRAPRG Elements	4.87-177
4.87.14.1	Integral Calculation for $q \geq 0$ and any p. (Function DKINT)	4.87-179
4.87.14.2	Integral Calculation for $p \geq 0$ and $q < -1$ (Function DK89)	4.87-179
4.87.14.3	Integral Calculation for $p < 0$ and $q < -1$ (Function DK100)	4.87-180
4.87.14.4	Integral Calculations for $p > -1$ and $q = -1$ (Function DKJAB)	4.87-181
4.87.14.5	Integral Calculations for $p > -1$ and $q = -1$ (Function DK219)	4.87-182
4.87.14.6	Integral Calculations for $p = -1$ and $q = -1$ (Function DK211)	4.87-182
4.88	DETERMINANT METHOD OF EIGENVALUE EXTRACTION	4.88-1
5.	NASTRAN - OPERATING SYSTEM INTERFACES	
5.1	INTRODUCTION	5.1-1
5.2	NASTRAN ON THE IBM 7094/7040(44) DCS (IBSYS)	5.2-1
5.2.1	Introduction	5.2-1
5.2.2	Input/Output	5.2-1
5.2.3	Link Switching	5.2-3
5.2.4	Overlay Considerations and Implementation of Open Core	5.2-3
5.2.5	Execution Deck Setup	5.2-5
5.2.6	Generation of the NASTRAN Executable System ("Subsysing")	5.2-8
5.2.7	Machine Dependent Routines	5.2-10
5.2.8	Flow Charts for the DCS Machine Language GINØ Routines	5.2-11
5.2.9	DCS Overlay Diagrams	5.2-30
5.3	NASTRAN ON THE IBM SYSTEM/360 OPERATING SYSTEM (ØS)	5.3-1
5.3.1	Introduction	5.3-1
5.3.2	Input/Output	5.3-2
5.3.3	Link Switching	5.3-2
5.3.4	Overlay Considerations and Open Core Implementation	5.3-3
5.3.5	Execution Deck Setup	5.3-4
5.3.6	Generation of the NASTRAN Executable System (SUBSYSING)	5.3-9

Section			Page No.
	5.3.7	Machine Dependent Routines	5.3-10
	5.3.8	Special Error Codes from NASTRAN on the System/360	5.3-11
5.4		NASTRAN ON THE UNIVAC 1108 (EXEC 8)	5.4-1
	5.4.1	Introduction	5.4-1
	5.4.2	Input/Output	5.4-1
	5.4.3	Link Switching	5.4-2
	5.4.4	Overlay Considerations and Implementation of Open Core	5.4-3
	5.4.5	Execution Deck Setup	5.4-5
	5.4.6	Generation of the NASTRAN Executable System (Subsysing)	5.4-7
	5.4.7	Machine Dependent Routines	5.4-8
5.5		NASTRAN ON THE CDC 6400/6600 (SCOPE 3)	5.5-1
	5.5.1	Introduction	5.5-1
	5.5.2	Input/Output	5.5-1
	5.5.3	Layout of Core Storage	5.5-4
	5.5.4	Execution Deck Setup	5.5-6
	5.5.5	Generation of the NASTRAN Executable System	5.5-9
	5.5.6	Machine Dependent Routines	5.5-10
5.6		THE CDC 6400/6600 LINKAGE EDITOR	5.6-1
	5.6.1	Introduction	5.6-1
	5.6.2	Preparing for Linkage Editor Processing	5.6-6
	5.6.3	Designing an Overlay Program	5.6-7
	5.6.4	Linkage Editor Control Statements	5.6-12
	5.6.5	Examples of Linkage Editor Processing	5.6-23
	5.6.6	Storage Requirements for the Linkage Editor	5.6-29
6.		MODIFICATIONS AND ADDITIONS TO NASTRAN	
6.1		INTRODUCTION	6.1-1
6.2		FØRTRAN IV LANGUAGE RESTRICTIONS	6.2-1
6.3		THE EXECUTIVE CONTROL DECK	6.3-1
	6.3.1	The NASTRAN Card	6.3-1
6.4		THE CASE CONTROL DECK	6.4-1

Section		Page No.
6.5	THE BULK DATA DECK	6.5-1
6.6	RIGID FORMATS	6.6-1
6.7	FUNCTIONAL MODULES	6.7-1
6.8	STRUCTURAL ELEMENTS	6.8-1
6.8.1	IFP (Input File Processor)	6.8-1
6.8.2	GP1, GP2, GP3, and TA1 (Geometry Processor and Table Assembler Modules)	6.8-1
6.8.3	PLTSET (Plot Set Definition Processor)	6.8-2
6.8.4	SMA1 (Structural Matrix Assembler - Phase 1)	6.8-3
6.8.5	SMA2 (Structural Matrix Assembler - Phase 2)	6.8-3
6.8.6	SSG1 (Static Solution Generator - Phase 1)	6.8-3
6.8.7	SDR2 (Stress Data Recovery - Phase 2)	6.8-3
6.8.8	DSMG1 (Differential Stiffness Matrix Generator - Phase 1)	6.8-4
6.8.9	PLA1 (Piecewise Linear Analysis - Phase 1)	6.8-5
6.8.10	PLA3 (Piecewise Linear Analysis - Phase 3)	6.8-6
6.8.11	PLA4 (Piecewise Linear Analysis - Phase 4)	6.8-6
6.8.12	ØFP (Output File Processor)	6.8-7
6.9	PRINTED OUTPUT	6.9-1
6.10	PLOTTER OUTPUT	6.10-1
6.10.1	Changes to the Plotter Software	6.10-1
6.10.2	Changes to the PLØT Module, the Structural Plotter	6.10-3
6.10.3	Changes to the XYPLØT Module, the XY Plotter	6.10-4
6.10.4	Changes to the SEEMAT Module, the Matrix Plotter	6.10-4
6.10.5	Use of the NASTRAN Plotter Software in a New Module	6.10-6
6.10.6	NASTRAN General Purpose Plotter	6.10-14
6.11	ADDITION OF A NEW LINK	6.11-1
6.11.1	Modules to Include	6.11-1
6.11.2	Addition of New Modules	6.11-1
6.11.3	Generation of a New Link Specification Table and a New Link Driver	6.11-2
6.11.4	Subsys the New Link	6.11-4
6.11.5	Increasing the Link Limit	6.11-4

1.1 PROGRAM OVERVIEW

1.1.1 Objectives

The NASTRAN program has been designed according to two classes of criteria. The first class relates to functional requirements for the solution of an extremely wide range of large and complex problems in structural analysis with high accuracy and computational efficiency. These criteria are achieved by developing and incorporating the most advanced mathematical models and computational algorithms that have been proven in practice. In particular, they are achieved by providing such features as the bandwidth-with-active-column technique in matrix decomposition; packing routines to take maximum advantage of matrix sparsity so as to conserve input/output time; highly stable and efficient algorithms for the solution of problems in eigenvalue analysis and transient response; and an elegant approach to modeling the effects of control systems and other nonstructural components.

The second class of criteria relates to the operational and organizational aspects of the program. These aspects are somewhat divorced from structural analysis itself; yet they are of equal importance in determining the usefulness and quality of the program. Chief among these criteria are:

1. Simplicity of problem input deck preparation.
2. Minimization of chances for human error in problem preparation.
3. Minimization of need for manual intervention during program execution.
4. Ease of program modification and extension to new functional capability.
5. Ease of program extension to new computer configurations and operating systems, and generality in ability to operate efficiently under a wide set of configuration capabilities.
6. Capability for step by step problem solution, without penalty of repeated problem set up.
7. Capability for problem restart following unplanned interruptions or problem preparation error.
8. Minimization of system overhead, in the three vital areas:
    a. Diversion of core storage from functional use in problem solution.

b. Diversion of auxiliary storage units from functional to system usage.

c. System housekeeping time for performing executive functions that do not directly further problem solution.

These criteria are achieved in NASTRAN through modular separation of functional capabilities, organized under an efficient, problem-independent Executive System.

This approach is absolutely essential for any complex multi-operation, multi-file application program such as NASTRAN. To see this, one must examine the implications of modularity in program organization.

Any application computer program provides a selection of computational sequences. These are controlled by the user through externally provided options and parameter values. Since no user will wish to observe the result of each calculation, these options also provide for the selection of the data to be output.

In addition to externally set options, internal switches whose setting depend upon tests performed during the calculations will control the computation sequences. There is, therefore, a natural separation of computations into functional blocks. The principal blocks are called functional modules; modules themselves of course may, and usually must, be further organized on a sub-modular basis.

Despite this separation, however, it is clear that modules cannot be completely independent, since they are all directed toward solution of the same general problem. In particular, they must intercommunicate data among themselves. The principal problem in organizing any application program, large or small, is designing the data interfaces between modules.

For small programs, the standard techniques are to communicate data via subroutine calling sequences and common data regions in core. For programs that handle larger amounts of data, auxiliary storage is used; however, strict specifications of the devices used and of the data record formats are usually imposed.

The penalty paid is that of "side effects". A change in a minor subroutine initiates a modification of the data interfaces that propagates through the entire program. When the program is small, these effects may not be serious. For a complex program like NASTRAN, however, they would be disastrous.

This problem has been solved in NASTRAN by a separation of system functions, performed by an Executive System, from problem solution functions, accomplished by modules separated strictly along functional lines. Each module is independent of all other modules in the sense that modification of a module, or addition of a new module, will not in general require modification of other modules. Even so, programming constraints on module development do exist but are minor. The essential restrictions are:

1. Modules may interface with other modules only through auxiliary storage files, as opposed to passing information between each other while in core.

2. Since the availability and allocation of auxiliary files for module execution interact with the execution of other modules, no module can specify or allocate files for its input or output data. All auxiliary storage allocation is reserved as an Executive function.

3. Modules operate as independent subprograms, and may not call, or be called by, other modules. They may be entered only from the Executive System.

4. Modules may interface with the Executive System through a parameter table that is maintained by the Executive System. User-specified options and parameters are communicated to modules in this way. The major line of communication is one-way, from user to Executive routine to module. However, in addition, an appreciable two way communication, from module back to executive routine (and therefore to other modules) is permitted via the parameter table.

5. Intra-module parameter communication is format-free in the sense that each module defines and orders its own local parameter set internally. Thus each module is independent of common data formatting by any other module.

No other constraints, except those imposed by the resident compilers and operating systems, are required for functional modules.

1.1.2 Program Organization

Because of the very large size of the NASTRAN program (more than 750 decks and 300 individual overlay segments), execution as one physical program was not possible. However, to meet the stated design objectives, it was required that NASTRAN appear to the resident operating system as one program.

A program structure evolved which is basically computer independent, although the way in which the code structure is supported varies across the computers.

The NASTRAN program is divided into a series of logical pieces called links. Each link contains its own root segment (the set of subprograms which is always core resident for that link) and its own complete overlay structure. Each link is capable of performing a predefined subset of NASTRAN operations. Communication between links occurs through computer files. Control of the sequence of execution of the links is performed entirely by the NASTRAN program and requires no operator intervention. As a result of this approach, a NASTRAN program execution appears to the resident operating system as a normal batch job to be processed in the batch stream. Detailed descriptions of the way in which the link structure is implemented on each computer are given in section 5.

1.2 NASTRAN EXECUTIVE SYSTEM

1.2.1 Introduction

The essential functions of the Executive System are:

1. Establish and control the sequence of module executions according to options specified by the user.

2. Establish, protect, and communicate values of parameters for each module.

3. Allocate system files to all data blocks (a data block designates a set of data, matrix or table, occupying a file) generated during program execution. A file is "allocated" to a data block, and a data block is "assigned" to a file. The general data block I/Ø routine (GINØ) and the data card conversion routines (XRCARD and RCARD) are considered Input/Output utilities and are discussed separately in section 1.6.

4. Maintain a full restart capability for restoring a program execution after either a scheduled or unscheduled interruption.

The Executive System is open-ended in the sense that it can accommodate an essentially unlimited number of functional modules, files, and parameters. Modification of the Executive System necessary for change, addition, or extension of functional modules is restricted to changes in entries in control tables stored within the Executive routines.

Program execution is divided into two phases: 1) the Preface, in which modules XCSA, IFP1, XSØRT, IFP and XGPI are executed to: a) process the NASTRAN input data deck and b) perform general problem initialization; and 2) the program body itself, in which the sequence of program operations is controlled by the Operation Sequence Control Array (ØSCAR) Executive table, which was developed in the XGPI module of the Preface. A diagram of a sample NASTRAN input data deck is shown in Figure 1. Note that a NASTRAN input data deck consists of 3 separate decks: 1) the Executive Control Deck, 2) the Case Control Deck and 3) the Bulk Data Deck. A detailed description of the contents of the NASTRAN data deck is given in section 2 of the User's Manual. The flow of operations during the Preface is presented in Figure 2. The numbers in the blocks in Figure 2 refer to section numbers where more detailed explanations of the subroutines and modules can be found.

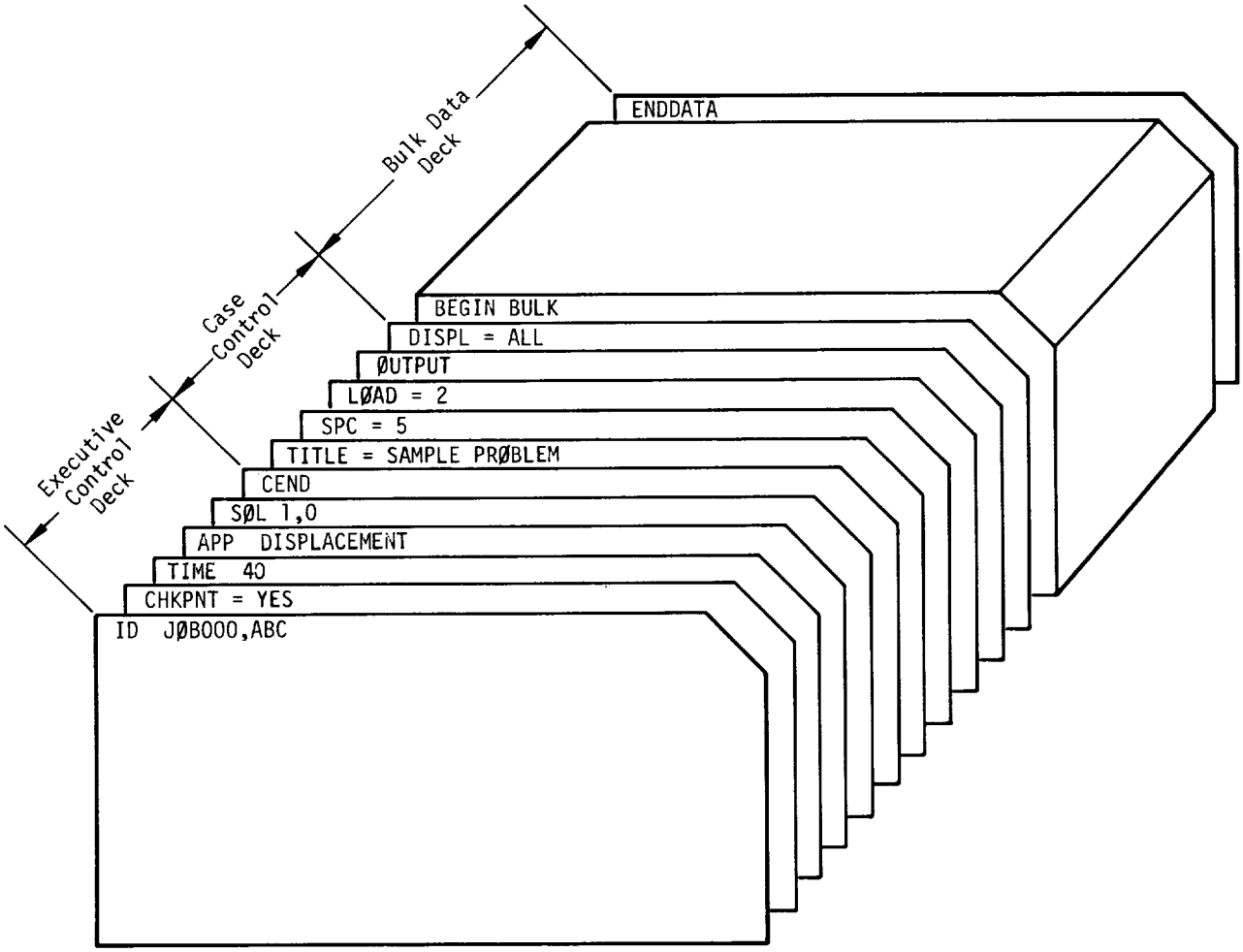


Figure 1. Sample NASTRAN input data deck.

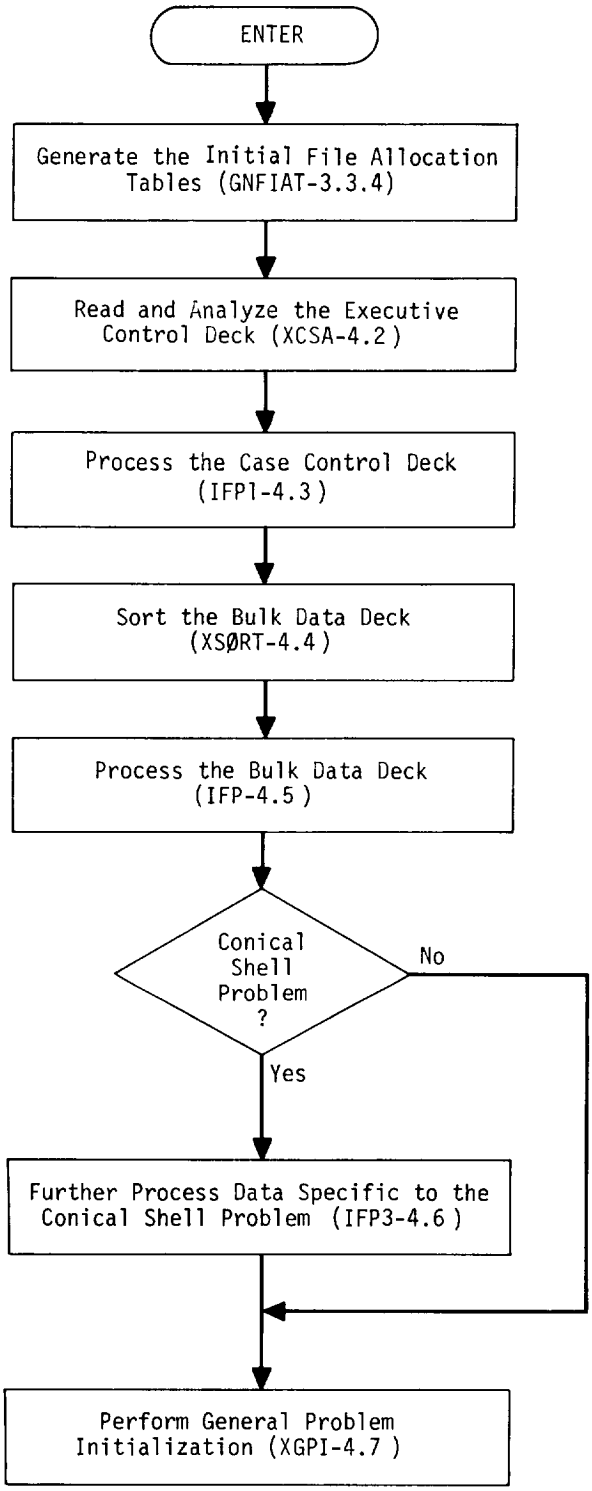


Figure 2. Flow of operations during the Preface.

1.2.2 Executive Operations During the Preface

The sequence of Preface operations shown in Figure 2 is controlled by the Sequence Monitor Initialization subroutine, SEMINT (see section 3.3.3). Each routine called by SEMINT is discussed in the following sections. The numbers in the section headings refer to section numbers where more detailed information on the subroutine or module can be found.

1.2.2.1 Generation of the Initial File Allocation Tables (GNFIAT section 3.3.4)

Two file allocation tables are maintained by the NASTRAN Executive System. One table, FIAT, (see section 2.4) defines the files to which data blocks generated during solution of the problem will be allocated. The second table, XFIAT, (see section 2.4) includes files to which permanent Executive data blocks, such as the New Problem Tape, the Old Problem Tape, plot tapes, and the User's Master File are assigned.

The New Problem Tape will contain those data blocks generated during the solution that are necessary for restarting the problem at any point. The Old Problem Tape contains the data blocks saved from some previous execution that may serve to bypass steps in the solution of the new problem. The User's Master File is a permanent collection of useful information, such as material properties, that may be used to generate input data.

The generation of the XFIAT and FIAT tables is a computer dependent operation since direct interface with the operating system of the computer must be made. The GNFIAT routine, which accomplishes this function, interrogates file tables in the nucleus of the operating system. Files which are available for use by the NASTRAN program are reserved, and the unit numbers for these files are stored in the NASTRAN file allocation tables. An indication of which units are physical tapes is also stored. If the number of files available is insufficient to run the problem, an error message is generated, and the run is aborted.

1.2.2.2 Analysis of the Executive Control Deck (XCSA Section 4.2)

The Executive Control Deck is processed and analyzed by the XCSA Executive Preface module. The Executive Control Deck includes cards which describe the nature and type of solution to be performed. This includes an identification of the problem, an estimated time for solution of the problem, the approach, a selection of the Rigid Format to be executed or an alternative sequence of NASTRAN operations (DMAP) to control the solution, a restart deck from a previous run if the

solution is to be restarted, an indication of any diagnostic printout to be made, a specification of whether the problem is to be checkpointed or not, and, if a Rigid Format is selected, any desired alterations to that format. Section 2 of the User's Manual should be consulted for the formats of, and restrictions on, each of the cards in the Executive Control Deck. The approach (APP) card, and the solution (SØL) card, which selects a particular solution (Rigid Format) to be executed, are worthy of special note. However, first some introductory definitions are required.

The sequence of operations to be executed during the program body is written in a data block oriented language called DMAP, an acronym for "Direct Matrix Abstraction Program". A DMAP instruction is a statement in the DMAP language, a DMAP sequence is a set of DMAP instructions, and a DMAP loop is a DMAP sequence to be repeated. A DMAP module is one which is "called" by means of a DMAP instruction.

A Rigid Format consists of: a) a fixed pre-stored DMAP sequence and b) its associated restart tables. A Rigid Format performs a specific (structural) problem solution. Section 3 of the User's Manual presents the DMAP sequence and the associated restart tables for each Rigid Format.

The APP card of the Executive Control Deck defines the problem solution approach. The APP card is required, and there are two options on the APP card: DISPLACEMENT or DMAP. The SØL card has the form

SØL n,m

where n = Rigid Format number, and m = a subset of the Rigid Format. The SØL card is required if the DISPLACEMENT option is chosen on the APP card. The SØL card must not be present in the deck if the DMAP option is chosen.

In addition to using the Rigid Formats provided automatically by NASTRAN, the user may wish either to execute a series of modules in a manner different from that provided by the Rigid Format, or to perform a series of matrix operations which are not contained in any existing Rigid Format. If the modifications to an existing Rigid Format are minor, the ALTER feature described in Section 2 of the User's Manual may be employed. Otherwise, a user-written Direct Matrix Abstraction Program (DMAP) should be used, in which case the card

APP DMAP

must be used. Chapter 5 of the User's Manual discusses DMAP.

Each of the cards comprising the Executive Control Deck is read via XRCARD (3.4.19) and analyzed. Depending on the card, information is either stored in various Executive tables maintained in core storage or written in the Executive Control Table (2.4.2.5) on the New Problem Tape for further processing during the general problem initialization phase (XGPI-4.7) of the Preface. Figure 3 presents the format of the Problem Tape. The formats of the New and the Old Problem Tapes are identical; only chronology defines their separate functions.

1.2.2.3 Processing of the Case Control Deck (IFP1 Section 4.3)

The Case Control Deck includes the following classes of cards: selection of specific sets of data from the Bulk Data Deck, selection of printed or punched output, definition of subcases, definition of structural plots to be made, and definition of XY plots to be made. Section 2 of the User's Manual discusses in detail all cards of the Case Control Deck.

This deck is read via XRCARD (3.4.19) and processed. Information defining set selection, output selection and subcase definition is written into the Case Control data block, CASECC. Information defining plot requests is written in the Plot Control (PCDB) and XY Control (XYCDB) data blocks.

If the problem is a restart, a comparison with the Case Control Deck from the previous run is made. Differences are noted in an Executive restart table, which is used in the general problem initialization phase (XGPI-4.7) of the Preface.

1.2.2.4 Sorting of the Bulk Data Deck (XSØRT Section 4.4)

The function of the XSØRT routine is to prepare a file on the New Problem Tape (see section 1.2.2.1) which contains the sorted Bulk Data Deck (bulk data). Operation of the routine is influenced by the type of run. If the run is a cold start, the bulk data is read from the system input file (e.g. card reader) or the User's Master File, sorted, and written on the New Problem Tape. If the run is an unmodified restart, (restarts are discussed in section 1.10), the bulk data is copied from the Old Problem Tape (see section 1.2.2.1) to the New Problem Tape. If the run is a modified restart, the bulk data is read from the Old Problem Tape, and cards are deleted and/or added in accordance with cards in the system input stream. The modified bulk data is sorted and written on the New Problem Tape. Additionally, any changes in the data are noted in the Executive restart table.

A printed list of the unsorted bulk data is given if requested by an ECHØ card in the Case Control Deck. Similarly, the sorted bulk data is echoed on request.

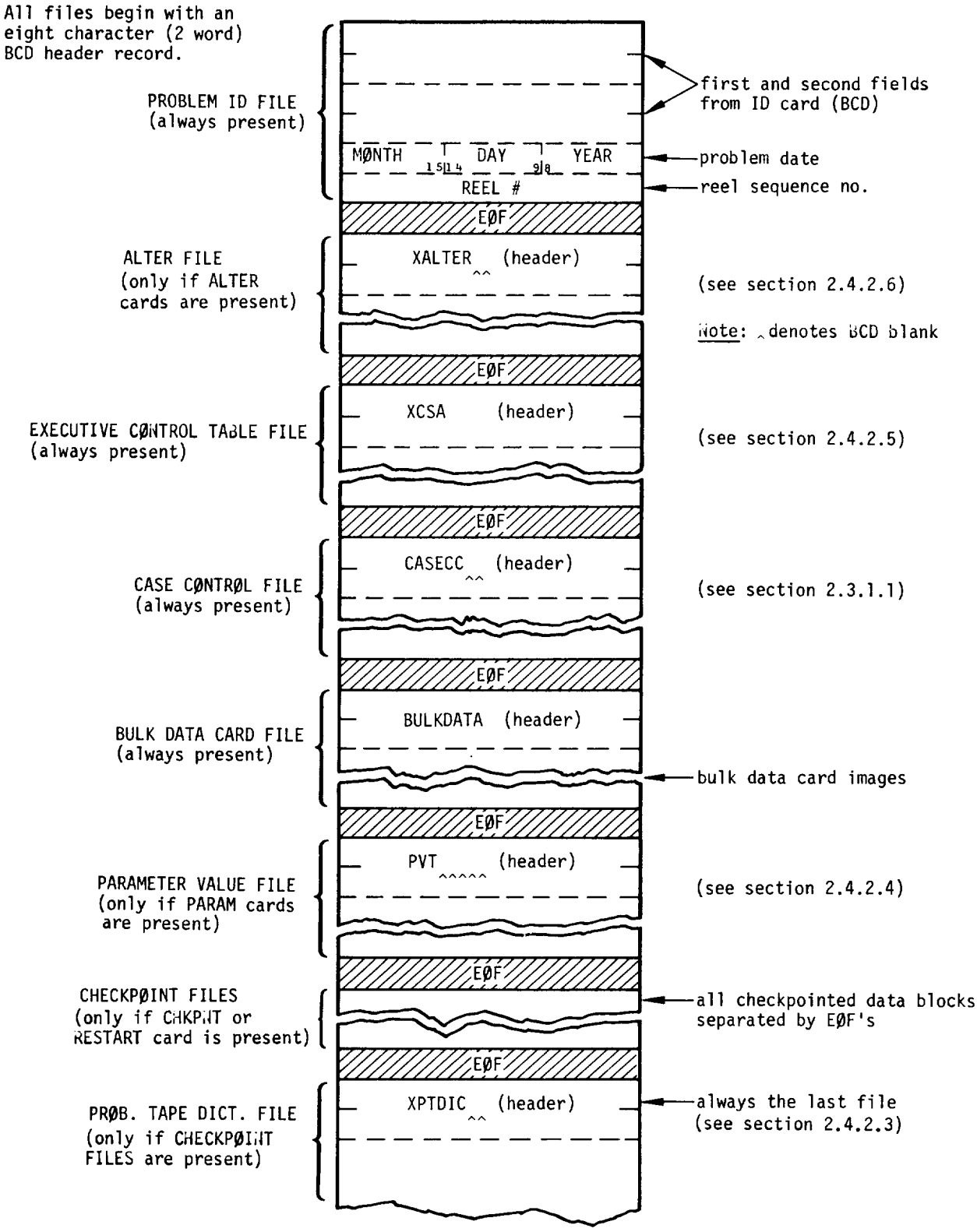


Figure 3. Problem tape format (same format for New Problem Tape and Old Problem Tape).

Since the collating sequence of alphanumeric characters varies from computer to computer, the sort routine converts all characters to an internal code prior to sorting. Following the sort, the characters are reconverted. In this way, the collating sequence is computer independent.

The algorithm used by the sort routine is biased toward the case where the data are in sort or nearly in sort. Consequently, Bulk Data Decks which are nearly in sort will be processed efficiently by the routine.

1.2.2.5 Processing of the Bulk Data Deck (IFP Section 4.5)

The sorted Bulk Data Deck is read card-by-card from the New Problem Tape by the Input File Processor (IFP) and converted to internal binary form by RCARD (3.4.20). Each of the cards is checked for correctness of format. If any data errors are detected, a message is written, and a switch is set to terminate the run at the conclusion of the Preface. Section 2 of the User's Manual presents a detailed description of all cards of the Bulk Data Deck.

Processing of each bulk data card depends on the type of card. All bulk data cards of the same type are written into the logical record to which the card type has been assigned. These records are organized into data blocks classified according to general categories of use and written on prescribed preallocated files.

1.2.2.6 Processing of Conical Shell Data (IFP3 Section 4.6)

If the problem is a conical shell problem, further processing of the bulk data specific to the conical shell problem is accomplished. The nature of this processing is to convert data for the conical shell model into formats of a conventional statics problem. The result is that the conical shell problem can be described in a format convenient to the analyst and processed by NASTRAN in a format convenient to the program.

1.2.2.7 General Problem Initialization (XGPI Section 4.7)

The Executive General Problem Initialization (XGPI) module is the heart of the Preface. Its principal function is to generate the Operation Sequence Control Array (ØSCAR-2.4.2.1), which defines the problem solution sequence. The ØSCAR consists of a sequence of entries, with each entry containing all of the information needed to execute one step of the problem solution. The ØSCAR is generated from information supplied by the user through his entries in the Executive Control Deck. This information is supplied by the SØL card, which points to a Rigid Format, or by a user supplied DMAP sequence.

The initial sequence of instructions was written in the Executive Control Table (2.4.2.5) on the New Problem Tape by the XCSA Preface module. This table is read to initiate assembly of the ØSCAR.

If the problem is a restart, the restart dictionary (contained in the Executive Control Table) and the Executive restart table are analyzed to determine which data blocks are needed to restart the solution and which operations in the ØSCAR need to be executed to complete the solution. Entries in the ØSCAR for operations not required for the current solution are flagged for no operation.

To aid in efficient assignment of data blocks to files, two attributes are computed and included with each data block in each entry of the ØSCAR. These attributes are: a) the ØSCAR sequence number when the data block is next used (NTU) and b) the ØSCAR sequence number when the data block is last used (LTU). Details of the file allocation are discussed in section 1.2.3.3.

When generation of the ØSCAR is complete, it is written on the Data Pool File (PØØL). If the problem is a restart, data blocks needed for the current solution are copied from the Old Problem Tape to the Data Pool File.

1.2.3 Executive Operations During Problem Solution

1.2.3.1 Sequence Monitor (XSEMi Section 3.3.7)

When the Preface has been completed, solution of the problem is initiated. This solution is controlled by the sequence monitor. Figure 4 shows the flow for the sequence monitor. Note that there are i copies of XSEMi within NASTRAN, one controlling each link's operation. Section 1.1.2 defined the necessity for these divisions.

The sequence monitor reads an entry from the ØSCAR (2.4.2.1) which defines one step in the problem solution in terms of: the operation to be performed, data blocks required for input, data blocks to be output, scratch files required and parameters used. The File Status Table (FIST-2.4.1.3), which relates the internal data block reference numbers (see section 1.6.4) to the file position in the File Allocation Table (FIAT-2.4.1.2), is created by the FIST generator, subroutine GNFIST. When the status table is complete, XSEMi moves the parameters required for the operation into blank common and calls the requested module (if within the current link) to begin the operation. If the requested module is not within the current link, ENDSYS (see section 3.3.5) is called and the Sequence Monitor within the new link is executed.

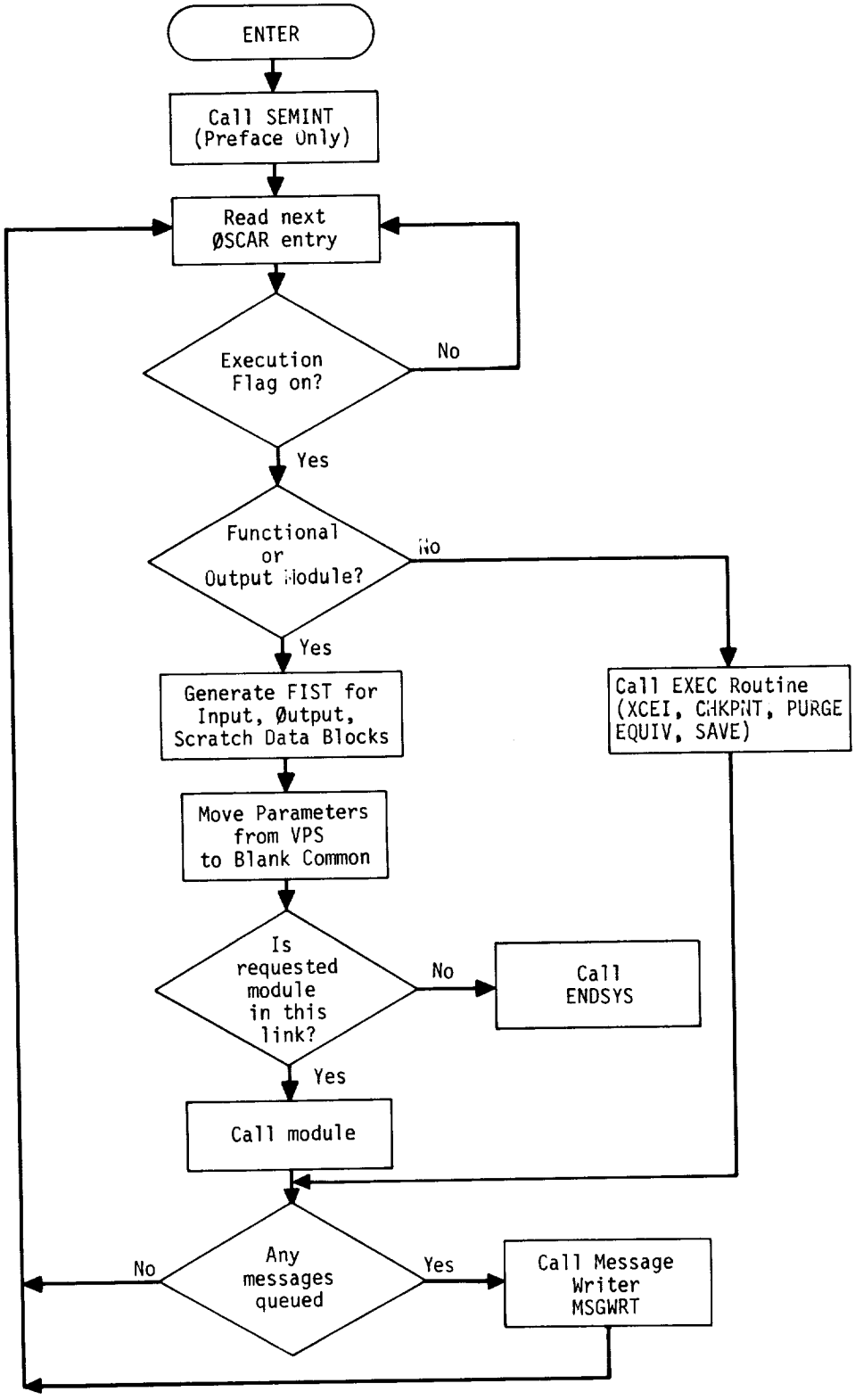


Figure 4. Flow diagram for the sequence monitor, XSEMi.

With the exception of XSFA, the seven routines described in the following subsections are Executive modules called directly by XSEMi to perform their specified functions.

1.2.3.2 FIST Generator (GNFIST Section 3.3.9)

The FIST generator, subroutine GNFIST, creates the File Status Table (FIST), which contains the linkage between the internal data block reference numbers and the actual system files listed in the File Allocation Table (FIAT). Each input, output and scratch data block required by the forthcoming module is assigned an internal reference number if found to be active in FIAT. A data block found to be inactive, that is purged or not generated, will not be assigned a reference number. This missing reference number will cause the accessing module to be signaled regarding the inactive status. If, during the generation of the FIST, a data block is not found in the FIAT, active or inactive, the Executive Segment File Allocator (XSFA) module is called by GNFIST to make a file available to the subject data block.

1.2.3.3 Segment File Allocator (XSFA Section 4.9)

The Executive Segment File Allocator (XSFA) module, which is called exclusively by GNFIST, is the administrative manager of data blocks for NASTRAN. Since, in general, the number of data blocks required for solution of a problem far exceeds the number of files available, assignment of data blocks to files is a critical operation for efficient execution of NASTRAN.

The Executive Segment File Allocator module is called whenever a data block is required for execution of an operation but is not currently assigned to a file, (i.e., does not appear in the FIAT). When the Segment File Allocator is called, it attempts to allocate not just for the data block initiating the call, but for as much of the remaining problem solution as possible. This allocation depends on the type of problem, the number of files available, and the range of use of the remaining data blocks.

The Segment File Allocator reads entries from the ØSCAR from the point of current operation to the end of the problem solution. The FIAT table entries are created in which attributes of the data blocks, including their next use (NTU) and last use (LTU), are stored. Data blocks which are currently assigned to files but are no longer required for problem solution are released. In certain cases, when the range of use of a data block is large, it may not be possible to allocate a file to the data block throughout its range of use. In this case, pooling of the

data block is required so that the file to which the data block was assigned may be freed for another allocation. The next time used (NTU) attribute for a data block is used to efficiently pool data blocks. In general, the data block whose next use is the furthest from the current point is pooled, that is, copied onto the Data Pool File (PØØL). The format of the Data Pool File is shown in Figure 5.

One additional check is made with regard to pooling. The operation of the Segment File Allocator itself is less expensive than a pooling operation. Therefore, pooling occurs only when the module for which the allocation was required cannot be allocated without pooling.

When the Segment File Allocator is complete, a new File Allocation Table (FIAT) has been generated. This table is used until the solution again reaches a point where a data block is required to execute an operation but is not assigned to a file.

1.2.3.4 Interpretation of Executive Control Entries (XCEI Sections 4.11, 4.12, 4.13, 4.14)

Executive control entries include the DMAP instructions: REPT, JUMP, CØND and EXIT. Executive control entries in the ØSCAR are processed by the Executive Control Entry Interpretor (XCEI). When such an entry is encountered in the ØSCAR, the Control Entry Interpretor is called by XSEMi. If the operation is a jump, conditional jump or repeat, the ØSCAR is repositioned accordingly. If the operation is an exit, the NASTRAN termination routine PEXIT (3.4.22) is called.

1.2.3.5 Checkpointing Data Blocks (CHKPNT Section 4.10)

The checkpoint module (DMAP name: CHKPNT; entry point name: XCHK) copies specified data blocks required for problem restart onto the New Problem Tape and makes appropriate entries in the restart dictionary. This dictionary is also punched onto cards as each new entry is made. Thus, in the event of any unscheduled problem interruption, a restart from the last checkpoint can be made using the Problem Tape and the restart dictionary from the interrupted run.

1.2.3.6 Purging a Data Block (PURGE Section 4.16)

The purge routine (DMAP name: PURGE; entry point name: XPURGE) flags data blocks so that they will not be assigned to physical files. This special status provides a means for logically suppressing a segment of processing steps requiring the data block. Thus, if the function of a

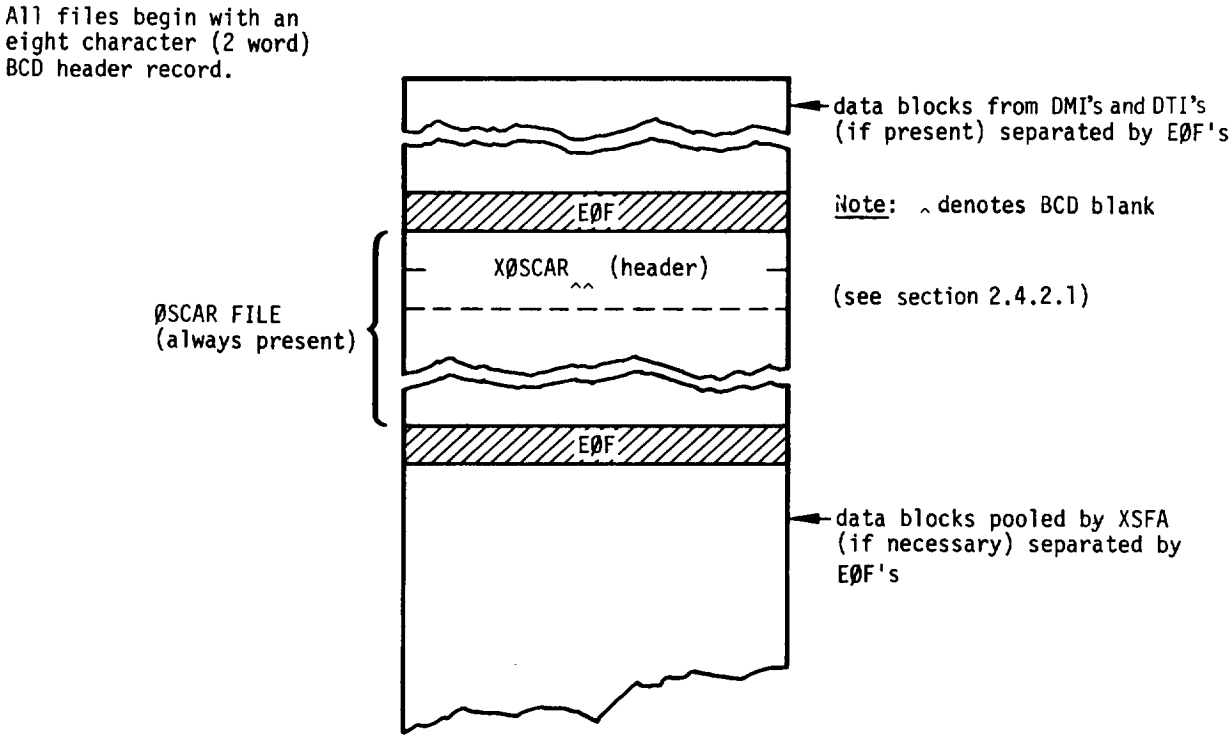


Figure 5. Format of the Data Pool File.

module is to multiply two matrices and add a third matrix to the product, the addition step might be deleted by purging the data block corresponding to the third matrix.

1.2.3.7 Equivalencing Data Blocks (EQUIV Section 4.17)

The equivalence routine (DMAP name: EQUIV; entry point name: XEQUIV) attaches one or more equivalent data block names to an existing data block. This special status provides a means of logically removing a module function by making a data block input to the module equivalent to a data block output from the module. Thus an entire module could be skipped, and an input data block "copied" to an output data block without physically moving the data from one file to another.

1.2.3.8 Saving Parameters (SAVE Section 4.15)

The save routine (DMAP name: SAVE; entry point name: XSAVE) provides a protection feature for the parameters communicated between, and used by, the functional modules. All variable parameters are stored within the VPS Executive table (see section 2.4). Prior to each module's operation, the subset of parameters required by the module is moved to blank common. The module may use or modify this subset of parameters as desired. When the module terminates operation, only those parameters within the subset designated to be saved are restored to the Executive table.

1.3 WORD SIZE AND COMPUTER HARDWARE CONSIDERATIONS

1.3.1 Introduction

Although NASTRAN is a FØRTRAN oriented system, considerable effort was required to develop programming and word handling techniques applicable to four separate computer configurations. These computers exhibit wide differences in their binary word sizes and integer representation method. The current computer configurations considered and their significant differences follow:

1. Computer - IBM 7094/7040 DCS

Word Size - 36 Bits

Character Capacity - 6 bits/character and 6 characters/word

Integer Representation - Sign and Magnitude

2. Computer - IBM System/360 series

Word Size - 32 Bits

Character Capacity - 8 bits/character and 4 characters/word (character ≡ byte)

Integer Representation - Twos complement for negative integers

3. Computer - UNIVAC 1108

Word Size - 36 Bits

Character Capacity - 6 bits/character and 6 characters/word

Integer Representation - Ones complement for negative integers

4. Computer - CDC 6600

Word Size - 60 Bits

Character Capacity - 6 bits/character and 10 characters/word.

Integer Representation - Ones complement for negative integers

Various Executive routines (e.g., XSØRT (4.4), XRCARD (3.4.19)) that deal directly with character strings from the input stream require some method of obtaining the above computer dependent information. Within the NASTRAN Preface, subroutine BTSTRP (3.3.2) solves an algorithm that determines which of the four computers is currently operating. This algorithm functions by inspecting the word length (by means of shifting and testing) and by checking the negative integer representation method. As a result of these tests, a word (MACH) within the SYSTEM Executive table (see section 2.4) is set to indicate the computer type. Since data within BTSTRP defines the number of bits-per-word (NBPW), the number of characters-per-word (NCPW), and the number of

bits-per-character (NBPC) for each computer type, the correct values for these parameters are also stored into the SYSTEM table. This table resides within the NASTRAN root segment and is thus accessable to any module or subroutine.

1.3.2 Alphanumeric Data

Data stored within a computer as binary-coded-decimal (BCD) characters must be represented by the proper hardware defined bit codes. These character codes (and in the case of the IBM System/360, the number of bits representing the code) vary among the NASTRAN computer types. Although the number of characters-per-word could have been obtained from the SYSTEM table, various data blocks and buffers within NASTRAN required firm entry sizes, regardless of computer type, to facilitate indexing. For these reasons, the minimum number of characters-per-word (4) among the four computer types was chosen as a program design standard. Computer types with a word capacity of greater than four characters will have the unused low order character positions filled with BCD blanks.

1.3.3 Word Packing

Standard FØRTRAN compilers do not provide the capability for storing or retrieving data that occupies less than a full computer word. Through the Machine Word Functions (MAPFNS, 3.4.1) routine some limited word packing (not to be confused with matrix packing) is performed within the Executive System and a few utility subroutines. Packing provides an efficient use of memory space at the expense of the additional operating time needed to combine or separate the elements of the packed words. The Machine Word Function ØRF is generally used for combining elements, while ANDF with a suitable mask is used for separating them.

1.3.3.1 Examples of Machine Word Functions (MAPFNS) Usage

Assume three 10-bit items of data occupy the low order 10 bits of three separate 30-bit computer words (A, B, and C). To pack these three items into a single 30-bit word (X), perform the following steps using the individual functions available within MAPFNS:

a) Left shift (LSHIFT) word A, twenty bits

b) Left shift (LSHIFT) word B, ten bits

c) Logically add (ØRF) words A and B; store into X

d) Logically add (ØRF) words X and C; store into X.

Assume two 8-bit items of data are packed into the left and right halves of a 16-bit word (X). To unpack these two items into the low order 8 bits of two separate 16-bit words (A and B), perform the following steps using the individual functions available within MAPFNS:

a) Create MASK containing 8 low order bits equal to 1 and the 8 high order bits equal to 0

b) Right shift (RSHIFT) word X, eight bits; store into A

c) Logically multiply (ANDF) word X by MASK; store into B.

In the preceding example, the word X remains unchanged since the functions return the requested modified result in a computer register.

1.4 SYSTEM BLOCK DATA SUBPROGRAM (SEMDBD)

NASTRAN contains a master block data program (SEMDBD) which is responsible for defining and initializing (root segment) common blocks. The common blocks referenced in SEMDBD are either Executive common blocks (XFIAT, XXFIAT, XFIST, etc.) which require initial values, or general information common blocks (SYSTEM, NAMES, TYPE, etc.) which are referenced by many modules. The source listing for SEMDBD identifies the common blocks, and it documents the data which are initialized, via comments. In addition, the Executive common blocks are documented in section 2.4 and the non-Executive common blocks in section 2.5. Certain parameters in these common blocks contain machine dependent values such as word size, number of BCD characters per word, etc. These values are set by subroutine BTSTRP (section 3.3.2) by identifying the machine on which the NASTRAN program is currently operating and setting the values accordingly.

1.5 THE OPEN CORE CONCEPT

1.5.1 Introduction

The design philosophy of the NASTRAN system dictated a completely open ended design whenever possible. NASTRAN was to have the flexibility to operate on a second generation machine with a 32K core (the IBM 7094/7040 DCS) as well as the largest of the IBM S/360 series of computers, and take complete advantage of the additional core storage without major program changes. The use of a fixed dimension for large arrays was outlawed since this automatically restricted the size of a problem that could be solved. Instead, modules were to be programmed to allocate space as required and to use spill logic to transfer data to scratch files if complete core allocation was impossible. In this manner, a problem might cause spill logic to be used on a computer with limited core storage, but not on a computer with a larger core storage capacity.

1.5.2 Definition of Open Core

The definition of open core is: a contiguous block of working storage defined by a labeled common block whose length is a variable determined by the NASTRAN Executive function CØRSZ. The implementation of this definition by the module writer consists of the origining of a labeled common block at the end of his overlay segment. This labeled common block contains a dimensioned variable of length 1. CØRSZ returns the number of words of core available between his open core origin and the end of core. The module writer can now write his program as if he had dimensioned his array by that number. In actuality, he is extending beyond the area reserved for the array into an area reserved for the job but not currently used by the segment. When implementing this concept, care must be taken to assure that the system does not use this area.

1.5.3 Example of an Application of Open Core

Figure 1 demonstrates the use of open core by two subroutines, A and B. By some means, which are machine dependent and are discussed in section 5, an end point is established for open core. The length of open core is then the difference between this end point and the labeled common block. In the example shown, subroutine A will have more open core available to it than B does.

```
      SUBRØUTINE A
      CØMMØN // XX
      CØMMØN /AX/ Z(1)
      INTEGER CØRSZ
      NZ = CØRSZ(Z(1),XX)
      DØ 10 I = 1, NZ
   10 Z(I) = I
      RETURN
      END
```

```
      SUBRØUTINE B
      CØMMØN // XX
      CØMMØN /BX/ Z(1)
      INTEGER CØRSZ
      NZ = CØRSZ(Z(1),XX)
      DØ 10 I = 1,NZ
   10 Z(I) = I
      •
      •
      •
      RETURN
      END
```

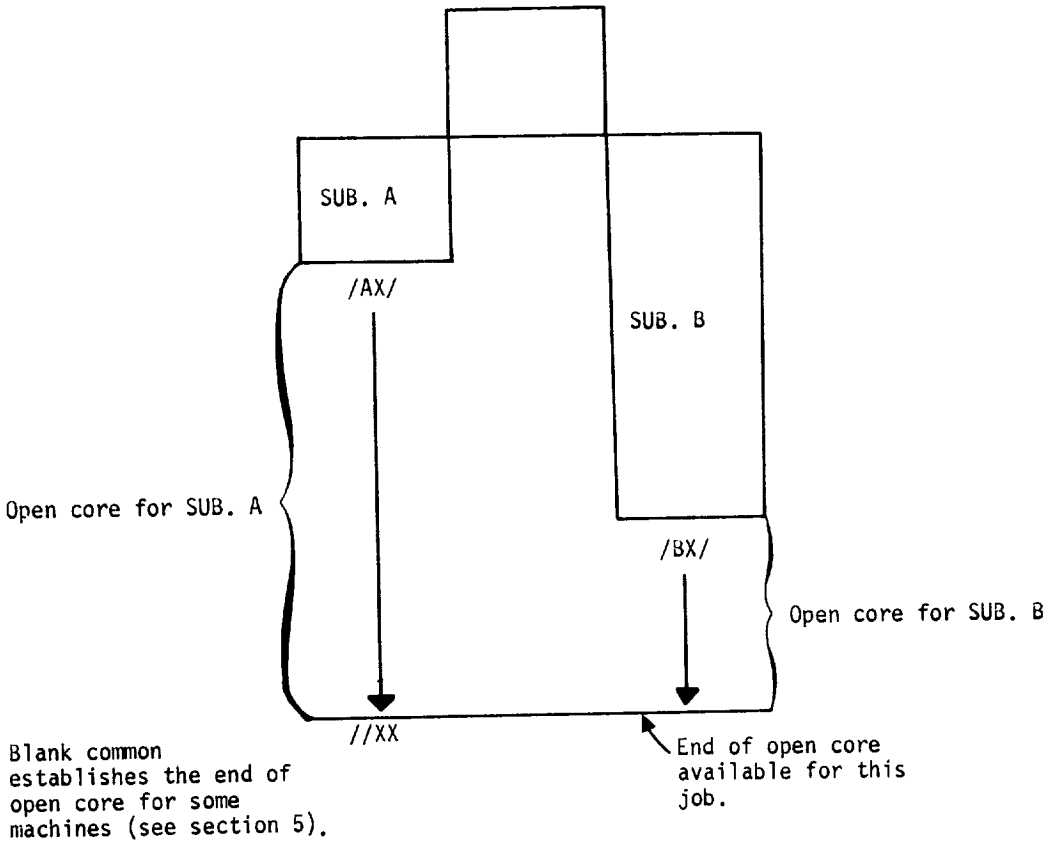


Figure 1. A example of the use of open core.

1.6 NASTRAN INPUT/OUTPUT

1.6.1 Introduction

The particular (IBM 7094, IBM S/360, Univac 1108, CDC 6600) operating system input and output files provide the required data connection between NASTRAN, the input data decks and the printed output. Utility subroutines XRCARD (section 3.4.19) and RCARD (section 3.4.20) convert special NASTRAN input card formats to standard FØRTRAN data words easily handled by all NASTRAN input processors. Printed output is generated through FØRTRAN formatted write statements. All internal data block input/output is handled by GINØ, the system of NASTRAN general purpose input/output routines. GINØ provides the required manipulation to tailor the variable length logical data records needed by most NASTRAN modules to fixed length records available on all direct access mass storage hardware.

1.6.2 Use of the Operating System Input File

The system input file is read only by the following routines within the NASTRAN Preface:

1. SEMINT (see section 3.3.3) reads the first card and processes it using utility XRCARD if it is the NASTRAN card (see section 6.3.1).

2. The Executive Control Deck containing free-field cards is read and processed by XCSA (section 4.2) using the XRCARD utility.

3. The Case Control Deck containing free-field cards is read and processed by IFP1 (section 4.3) using the XRCARD utility.

4. The Bulk Data Deck containing fixed-field cards is read by XSØRT (section 4.4). This data is subsequently processed by IFP (section 4.5) using the RCARD utility.

These card conversion utilities (XRCARD and RCARD) provide respectively all the free-field and fixed-field data card processing required by NASTRAN.

1.6.2.1 Use of the Subroutine XRCARD (See Section 3.4.19)

XRCARD interprets NASTRAN free-field data cards and processes the fields into a sequential buffer that can be easily handled by subsequent modules. Free-field data consist of series of data items separated by suitable delimiters and punched in non-specific card columns. Data items may include alphanumeric, integer, and various types of real variables. Field delimiters

may include the comma, slash, parenthesis, and blanks. For details regarding data and delimiter usage and the format of the sequential output buffer, see the XRCARD subroutine description in section 3.4.19.

1.6.2.2 Use of the Subroutine RCARD (See Section 3.4.20)

RCARD interprets NASTRAN fixed-field data cards and processes the fields into a sequential buffer that can be easily handled by subsequent modules. Fixed-field data consist of data items punched within specific card fields. Each eighty-column card is divided into an eight-column ID field (for the card mnemonic) followed by either eight eight-column data fields or four sixteen-column data fields. A special character (asterisk or plus) within the ID field determines when the card is to be interpreted as containing sixteen-column fields. The last eight columns of the card are for continuation mnemonics used by XSØRT and are not processed by RCARD. The data item within the ID field must be alphanumeric. The data items within all other fields may include alphanumeric, integer, and various types of real variables. For details regarding data and the format of the sequential output buffer, see the RCARD subroutine description in section 3.4.20.

1.6.3 Use of the Operating System Output File

Although NASTRAN printed output is formed and placed onto the system output file through use of standard FØRTRAN formatted write statements, two basic NASTRAN design concepts prohibit every operating module from generating printed output. Firstly, since the FØRTRAN I/Ø package for output generation occupies a sizable block of computer memory, this package is generally positioned by loader directives within specific output oriented segments, rather than within the root segment of the overlay, to reduce the total memory requirement. Secondly, because many functional modules generate the same or similar diagnostic and information messages, a NASTRAN message writer (MSGWRT) was developed to centralize message text and thus prevent duplications within many separate modules.

For the reasons previously discussed, NASTRAN output generation is restricted to a specific class of modules which can reside within an output oriented segment below the link root segment. These segments will contain the output producing modules such as the Output File Processor (ØFP- section 4.70), the Message Writer (MSGWRT - section 3.4.26), and the various table and matrix printers (TABPT - section 3.4.29, MATPRT - section 4.71, etc.) along with the output titling (PAGE - section 3.4.24) and necessary FØRTRAN I/Ø packages.

1.6.4 GINØ

GINØ is a collection of subroutines which provide for all input and output operations within NASTRAN except reading data from the resident system input file and writing data on the resident system output and punch files. These latter operations are accomplished through FØRTRAN formatted read/write statements. NASTRAN programs perform input/output operations by making the following calls to GINØ entry points:

1. ØPEN (see section 3.4.2)

   ØPEN initiates activity for a file (unless the data block assigned to the file is purged, in which case an alternate return is given). A working storage area (GINØ buffer), for use by GINØ, is assigned (allocated) by the calling program thus providing optimum allocation of storage by the calling program. This working storage area is reserved for use by GINØ until activity on the file is terminated by a call to CLØSE (see paragraph 4 below). On each of the NASTRAN computers, the working storage area allocated by the calling program is separate from the actual physical I/Ø buffer.

2. WRITE (see section 3.4.3)

   WRITE writes a specified (by the calling program) number of words on a file. The block of words to be written may comprise an entire logical record or portion of a logical record.

3. READ (see section 3.4.5)

   READ returns to the calling program a specified (by the calling program) number of words from the logical record at which the file is currently positioned. READ may be used to transmit an entire logical record or portion of a logical record.

4. CLØSE (see section 3.4.4)

   CLØSE terminates activity for a file. The working storage area assigned at ØPEN is released. The file is repositioned to the load point if requested.

5. REWIND (see section 3.4.8)

   REWIND repositions the requested file to the load point. The file must be "open", i.e. a REWIND operation is requested subsequent to a call to ØPEN and prior to a call to CLØSE.

6. FWDREC (see section 3.4.6)

   FWDREC repositions the requested file one logical record forward.

7. BCKREC (see section 3.4.7)

BCKREC repositions the requested file one logical record backwards.

8. SKPFIL (see section 3.4.10)

SKPFIL repositions the requested file forward or backward N logical files where N is specified by the calling program.

9. EØF (see section 3.4.9)

EØF writes a logical end-of-file on the requested file.

The basic unit of I/Ø in NASTRAN is a logical record. The length of a logical record is completely variable and may range from one word to an arbitrarily large number of words. For NASTRAN matrix data blocks, the convention was adopted that each column of the matrix would comprise one logical record. For NASTRAN data blocks containing tables, no rigid convention exists. Typically each logical record contains one table of a specific type.

The logical record concept provides greatest ease in programming. However, since these records must be stored on a physical device such as a drum, disk or tape, the characteristics of the device must be taken into consideration. The bulk of NASTRAN data is stored on drums or disks. For both these devices the common unit of organization is a track, which stores a fixed number of words. Thus, there is a conflict between the variable length GINØ records and the fixed length tracks.

This conflict is resolved by blocking. GINØ acts as the interface between the device and the NASTRAN program. Using this technique, the program itself need not be concerned with device considerations (which would create machine dependent code). GINØ has been parameterized so that different devices may be easily accommodated.

Basically, blocking provides for the reading and writing of fixed-length blocks. The length of a block is a function of the device. It may be one track, one-half track or other integral division of a track (but never more than one track). Because of the relatively large time to access a given position on a track (due to the rotational speed of the device and/or mechanical movement of the head to the track), a block size equal to one full track is the most desirable. However, limitations in the amount of storage available to hold the blocks in core is a second consideration.

Since logical record lengths are variable but the length of records physically read or written is fixed, logic must be provided to accommodate this situation. This logic is provided in the GINØ routine, which allows for the following cases:

1. Multiple logical records per block
2. Multiple blocks per logical record

The method by which physical input and output of blocks is accomplished by GINØ is machine dependent. On the IBM 7094, IØEX is used. On the IBM S/360, FØRTRAN binary read/write statements are used. On the Univac 1108, NTRAN is used. On the CDC 6600, SCØPE macros are used. These implementation differences are transparent to the NASTRAN applications programmer (functional module writer). The systems programmer who is interested in implementation details on the various machines is referred to section 5.

1.6.4.1 GINØ File Names

The names of files input as arguments to the GINØ routines listed above may be alphabetic (BCD, of the form 4HXXXX) or integer.

A GINØ file name is BCD if the file contains permanent Executive tables or data blocks. A list of these files for a particular NASTRAN run resides in the permanent portion of the FIST Executive table. The following list presents all current Executive files with their BCD file names:

File	BCD File Name
Data Pool File	PØØL
New Problem Tape	NPTP
Old Problem Tape	ØPTP
BCD Plot Tape	PLT1
Binary Plot Tape	PLT2
User's Master File	UMF
New User's Master File	NUMF
User Input File	INPT .

Functional modules should not access these permanent Executive files. Functional modules access the files on which their input, output and scratch data blocks reside by internal integer GINØ file names. Prior to calling a functional module, the link driver routine, XSEMi, calls subroutine GNFIST (GNFIST is called exclusively by XSEMi) to generate the FIST Executive table. For each input, output or scratch data block required for operation of a module (this information being contained in the ØSCAR entry), GNFIST searches the FIAT to find the data block. If the data block is in the FIAT and a file has been assigned to it, an internal GINØ file number denoting the data block and a pointer (index) to the entry in the FIAT is placed in the FIST. The following convention is used for internal GINØ file numbers: input data blocks -- 100 + position in the ØSCAR entry; output data blocks -- 200 + position in the ØSCAR entry; scratch data blocks -- 301 through 300 + n where n = number of scratch data blocks as defined in the MPL. (The position in the ØSCAR entry is the position in the DMAP instruction). If the data block is in the FIAT and is purged, no entry is placed in the FIST. For example, consider the following DMAP calling sequence for functional module XYZ:

```
XYZ      A,B,C/D,E,F,G/V,N,PARM1/V,N,PARM2 $
```

The data blocks input to the module are A, B and C; the data blocks output from the module are D, E, F and G; the module's parameters are PARM1 and PARM2. Note that internal scratch files are not mentioned in the DMAP calling sequence. The number of scratch files for a module is defined in the Module Property List (MPL) Executive table (see section 2.4) and is communicated to the Executive System via the ØSCAR. Details on the syntactical rules of DMAP are given in section 5 of the User's Manual.

In order to read the input data block B, the GINØ file number internal to XYZ is 102; in order to write data block D, the GINØ file number is 201. The third of, say, five scratch data blocks is referenced by XYZ through the GINØ file number 303.

1.7 NASTRAN MATRIX ROUTINES

1.7.1 Introduction

The requirement that NASTRAN handle large structural analysis problems implies that NASTRAN should be able to manipulate and store large matrices efficiently and effectively. In general, the matrices generated in the displacement approach tend to be sparse (i.e., the number of non-zero terms in any column of a matrix is small compared to the order of the matrix). The NASTRAN matrix routines, ADD, MPYAD, DECØMP, etc., which are described in section 3.5, are optimized as much as possible to take advantage of matrix sparsity and thus eliminate many unnecessary operation on zero elements. In order to aid in these operations and to make effective use of auxiliary storage, a packing scheme was devised to store only the non-zero terms in a column.

1.7.2 Matrix Packing and Unpacking

The need for a matrix packing routine can be seen by computing the auxiliary storage required to hold a 10,000 order matrix which is 1% dense (i.e., the average number of non-zero terms in a column is 100). With no packing technique, 10^8 words of storage are required to hold the matrix. Using the NASTRAN packing routines, a maximum of 2×10^6 words of storage are required if the terms are scattered, and 10^6 words are required if the terms occur in a band.

The subroutines BLDPK, INTPK, PACK, and UNPACK, along with their additional entry points, provide the matrix packing/unpacking capability of NASTRAN. The user should refer to the descriptions of these subroutines in sections 3.5.1 through 3.5.4 for a detailed description of the packing logic.

Matrices are stored by columns, and subroutines PACK/UNPACK provide the ability to pack/unpack a complete column. The capability is also provided to pack/unpack a column from the first non-zero element to the last.

An added feature of the packing routines is that subroutines BLDPK and INTPK provide the capability of packing/unpacking one element at a time. By use of INTPK, a matrix can be read element-by-element, such that an entire matrix can be processed without any appreciable core storage requirements. Likewise, by using BLDPK, a matrix can be built one element at a time. This is an extremely important feature to routines that must process matrices when storage is limited.

1.7.3 The Nested Vector Set Concept Used to Represent Components of Displacement

In constructing the matrices used in the Displacement Approach, each row and/or column of a matrix is associated closely with a grid point, a scalar point or an extra point. Every grid point has 6 degrees of freedom associated with it, and hence 6 rows and/or columns of the matrix. Scalar and extra points only have one degree of freedom. At each point (grid, scalar, extra) these degrees of freedom can be further classified into subsets, depending on the constraints or handling required for particular degrees of freedom. (For example in a two-dimensional problem all "z" degrees of freedom are constrained and hence belongs to the s (single-point constraint) set). Each degree of freedom can be considered as a "point", and the entire model is the collection of these one-dimensional points.

Nearly all of the matrix operations in displacement analysis are concerned with partitioning, merging, and transforming matrix arrays from one subset of displacement components to another. All the components of displacement of a given type (such as all points constrained by single-point constraints) form a vector set that is distinguished by a subscript from other sets. A given component of displacement can belong to several vector sets. The mutually exclusive vector sets, the sum of whose members are the set of all physical components of displacements, are as follows:

u_m points eliminated by multipoint constraints,

u_s points eliminated by single-point constraints,

u_o points omitted by structural matrix partitioning,

u_r points to which determinate reactions are applied in static analysis,

u_ℓ the remaining structural points used in static analysis (points left over),

u_e extra degrees of freedom introduced in dynamic analysis to describe control systems etc.

The vector sets obtained by combining two or more of the above sets are (+ sign indicates the union of two sets):

$u_a = u_r + u_\ell$, the set used in real eigenvalue analysis,

$u_d = u_a + u_e$, the set used in dynamic analysis by the direct method,

$u_f = u_a + u_o$, unconstrained (free) structural points,

$u_n = u_f + u_s$, all structural points nct constrained by multipoint constraints,

$u_g = u_n + u_m$, all structural (grid) points including scalar points,

$u_p = u_g + u_e$, all <u>physical</u> points.

In dynamic analysis, additional vector sets are obtained by a modal transformation derived from real eigenvalue analysis of the set u_a. These are:

ξ_o rigid body (<u>zero</u> frequency) modal coordinates,

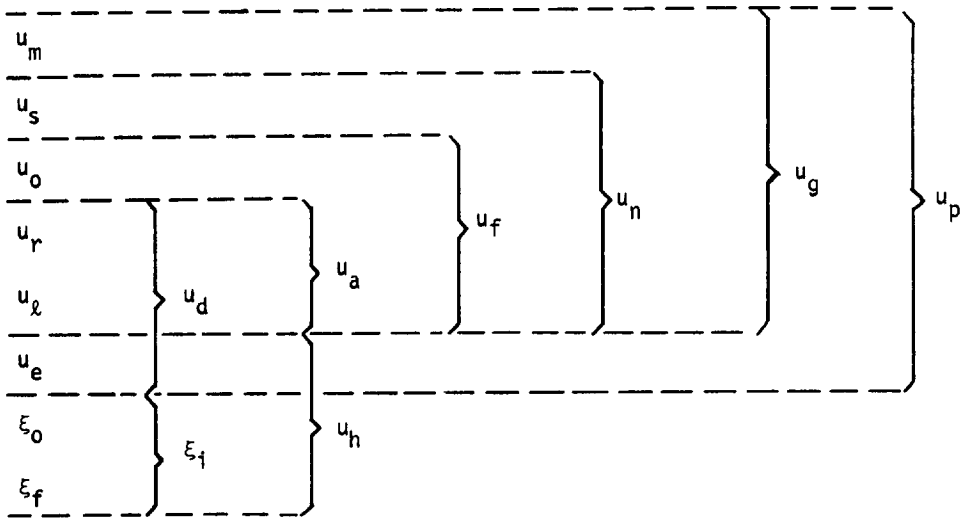
ξ_f <u>finite</u> frequency modal coordinates,

$\xi_i = \xi_o + \xi_f$, the set of all modal coordinates.

One vector set is defined that combines physical and modal coordinates. That set is

$u_h = \xi_i + u_e$, the set used in dynamic analysis by the modal method.

The nesting of vector sets is depicted by the following diagram:



The data block USET (USETD in dynamics) is central to this set classification. Each word of USET corresponds to a degree of freedom in the problem. Each set is assigned a bit in the word. If a degree of freedom belongs to a given set, the corresponding bit is on. Every degree of freedom can then be classified by analysis of USET. The common block /BITPØS/ relates the sets to bit numbers.

Two types of operations occur repeatedly. The first is the partitioning or sort operation. Examples are:

$$\{u_g\} \Longrightarrow \begin{Bmatrix} u_n \\ \hline u_m \end{Bmatrix} \quad ; \tag{1}$$

and

$$\left[K_{nn}\right] \Longrightarrow \left[\begin{array}{c:c} K_{ff} & K_{fs} \\ \hline K_{sf} & K_{ss} \end{array}\right] . \tag{2}$$

The second is the recombining (or merge) operation:

$$\{u_g\} \Longleftarrow \begin{Bmatrix} u_n \\ \hline u_m \end{Bmatrix} . \tag{3}$$

These operations can be completely described by a "partitioning" vector whose length corresponds to the length of the major set (the u_g set in Equation 1) and whose elements are zeros or ones depending on whether a degree of freedom belongs to the upper (the u_n set in Equation 1) subset or the lower (the u_m set in Equation 1) subset. Such a partitioning vector can be constructed by subroutine CALCV, which is described in section 3.5.5. This operation is described throughout the documentation by the notation USET (UG,UN,UM) where UG (u_g) is the major set, UN (u_n) is the zero set, and UM (u_m) is the one set. The partitioning vector generated by subroutine CALCV is input to the matrix routine PARTN (section 3.5.6) to perform operations similar to those in Equations 1 and 2 and is input to the matrix routine MERGE (section 3.5.6) to perform operations similar to that in Equation 3.

1.8 GENERATION OF MATRICES

The Structural Matrix Assembler (SMA) modules generate the stiffness, structural damping, mass and damping matrices for the structural model. SMA1 generates the stiffness matrix exclusive of general elements, $[K^{x}_{gg}]$, and the structural damping matrix, $[K^{4}_{gg}]$; SMA2 generates the mass matrix, $[M_{gg}]$, and the damping matrix, $[B_{gg}]$; and SMA3 generates the final stiffness matrix, $[K_{gg}]$, by generating a matrix for each general element in the model, and successively adding these matrices to $[K^{x}_{gg}]$. Other matrix generation modules are: 1) DSMG1, which generates the differential stiffness matrix, $[K^{d}_{gg}]$, for use in the Static Analysis with Differential Stiffness Rigid Format and in the Buckling Analysis Rigid Format; 2) PLA1, which generates the stiffness matrix for linear elements, $[K^{\ell}_{gg}]$, for use in the Piecewise Linear Analysis Rigid Format; 3) PLA4, which generates the stiffness matrix for nonlinear elements, $[K^{n\ell}_{gg}]$, for use in the Piecewise Linear Analysis Rigid Format; 4) MTRXIN, which provides a two-fold capability: a) to provide for direct input matrices such as control systems in the dynamics Rigid Formats, and b) to provide the DMAP user a capability of converting matrices input on DMIG bulk data cards to NASTRAN matrix format; and 5) the IFP module which provides the user the capability of converting matrices input on DMI bulk data cards to NASTRAN matrix format. Detailed information on each of these modules can be found in section 4, Module Functional Descriptions. The central role that the ECPT data block plays in the formation of the structural matrices generated in modules SMA1, SMA2, DSMG1, PLA1 and PLA4 is explained in the following subsections.

1.8.1 The ECPT Data Block

NASTRAN embodies a lumped element approach, i.e., the distributed physical properties of a structure are represented by a model consisting of a finite number of idealized substructures or elements that are interconnected at a finite number of points. An element will affect terms in the matrices only in rows and columns related to its interconnected points. Hence each column of these matrices may be formed using only elements connected to the grid or scalar point associated with that column.

The Table Assembler (TA1) module constructs the Element Connection and Properties Table (ECPT) data block for use in the generation of these structural matrices. Each record of the ECPT corresponds to a grid point or a scalar point of the model, and, conversely, every grid point or scalar point of the model corresponds to a record of the ECPT. The point to which a

record of the ECPT corresponds is called the pivot point of the record. Each record contains the connection and properties data for all elements connected to the pivot point. Hence data for an element will appear n times, where n is the number of points defining the element.

1.8.2 Structural Elements

The basis for the structural matrices in NASTRAN are the finite structural and scalar elements. Each element generates matrix terms connecting and connected to the grid and scalar points given on its input connection card (e.g., CRØD). A structural element generates 6 by 6 matrix partitions related to the six degrees of freedom of each connecting grid point. A scalar element generates one term for each connection.

The stiffness matrix, [K], for a structural element consists of a 6 by 6 partition for each combination of the connected grid points. For example, a BAR or RØD element is connected to two grid points, "a" and "b". The stiffness matrix partitions are: $[K_{aa}]$, $[K_{ab}]$, $[K_{ba}]$ and $[K_{bb}]$. A triangular element (e.g., TRMEM) is connected to three points. It will generate nine partitions: $[K_{aa}]$, $[K_{ab}]$, $[K_{ac}]$, $[K_{ba}]$, $[K_{bb}]$, $[K_{bc}]$, $[K_{ca}]$, $[K_{cb}]$ and $[K_{cc}]$.

Although the actual equations for the element stiffness, mass and damping matrices are different for each element, the solutions follow a definite pattern. The element connection, orientation and property data are given in the ECPT data block. The coordinate system data for orienting the global coordinates at each grid point are given in the CSTM data block. The material properties are given in the MPT and DIT data blocks. A utility routine, PRETRD, is available to fetch coordinate system data, and a utility routine, PREMAT, is available to fetch material properties.

1. An element coordinate system is calculated using the locations of the grid points. Using these data a matrix, [E], is formed, which transforms displacements from element coordinates to basic coordinates.

2. The stiffness matrix may be formed in element coordinates using many methods. For the simple elements (e.g., RØD) the terms are direct functions of the geometry, properties and material coefficients of the element. For some elements the matrix is first formulated in terms of generalized coordinates, {q}, usually the coefficients of a power series. In general coordinates, the matrix is $[K_q]$. Transformation matrices, $[H_i]$, are generated to

transform the displacements at the grid points in element coordinates {u}, to the general coordinates {q}.

3. The global coordinate system orientation matrix, [T], for each grid point is calculated.

4. The stiffness matrix partition for the columns related to point j and the rows related to point i is $[K_{ij}]$. In general it is formed by the equation

$$[K_{ij}] = [T_i]^T[E][H_i]^T[K_q][H_j][E]^T[T_j]. \quad (1)$$

In order to generate a particular partition, $[K_{ij}]$, it is often necessary to generate [K]. Only those partitions $[K_{ij}]$, where i is the pivot point and j = 1,2,...,n (n being the number of grid points associated with the element), are useful for the current ECPT record being processed. The unused partitions are recalculated and used when $j \neq i$ appears as a pivot point in a subsequent ECPT record. An alternate procedure for matrix generation, which is not used, would be to calculate all of the element matrices once and store them on an auxiliary file for future use when needed. The alternate procedure is less efficient for large problems, where efficiency really counts, because the recalculation time is less than the time required to recover element matrices from the auxiliary file.

1.9 TERMINATION PHILOSOPHY AND DIAGNOSTIC MESSAGES

Certain restrictions are placed upon the functional module writer with regard to run termination and error diagnostics. This is necessary in order to complete certain functions upon terminating and to maintain uniformity with regard to diagnostic messages.

A functional module writer is required to utilize a message writer (MSGWRT, section 3.4.26) to print all of his messages. In this manner similar message formats do not have to be duplicated in each module. Also, in order to avoid placing the I/Ø conversion routines and the lengthy format statements in the root segment, the message writer is restricted to its own overlay segment, Communication between the module and the message writer is via a queued message concept. Subroutine MESAGE (section 3.4.25) is called to store the message parameters. In the case of a fatal message, a dump is taken if a DIAG 1 card is present in the Executive Control Deck, and PEXIT (section 3.4.22) is called. For non-fatal messages, the message is queued and control given back to the user. The message queue is printed after each module is executed.

In order for any routine to terminate the current execution, a call to PEXIT must be made. PEXIT handles all the functions necessary to wrap up the run: flushing output buffers, printing queued messages, and punching the last card of the checkpoint dictionary.

1.10 RESTARTS IN NASTRAN

NASTRAN is designed to run large problems with long running times. Even with the best of computer systems, a hardware, operator, or system failure is not uncommon for long running jobs. At the same time, the large volumes of data and the complexity of the structures that can be modeled and analyzed using NASTRAN make it highly likely that user input data errors will occur. Many of these errors are of a subtle type, meaning that they cannot be immediately detected in the NASTRAN Preface by the modules which process the data decks. To deal with these problems, and to save machine time on runs which abort because of data or system errors, NASTRAN has a sophisticated checkpoint and restart capability (see section 3 of the User's Manual for a discussion of restarts from the user point of view). The overall design philosophy for restart is twofold. A restart selectively executes only the modules necessary to accomplish a user-input data change. The user is able to change any part of his problem including structural model changes, additional cases, or more output requests. At the same time restarts are automatic as far as user interference is concerned. The user need only checkpoint (see section 1.2.3.5) his original run and submit changes to the original run on subsequent runs. The user does not have to analyze the effect of his changes. In addition the selective nature of restart allows the program to proceed with implicit errors (errors present in the data but not yet identified) until no further valid progress can be made. The work accomplished to this point is not lost, but rather only the table or matrix data block in error must be corrected to allow the program to proceed. Much error checking can be deferred until the actual module using the data is in control. The remainder of this section will explain the program mechanics by which restart is accomplished.

In NASTRAN there are four general types of restarts. Unmodified Restart (UMR), Psuedo Modified Restart (PMR), Modified Restart (MR), and Rigid Format Switch (RFS). Note that in the User's Manual UMR's and PMR's are described together as Unmodifed Restarts. These classifications are for descriptive and internal purposes. The user need not know anything about which type is which. The basic characteristics of each type will be described below. An Unmodified Restart results when the user simply resubmits a problem with no data changes. It is used to continue a solution from the point of interruption. Presumably the problem aborted previously due to time expiring, machine error, system error, etc. The restart dictionary (created while checkpointing) is processed, and the solution is started again at the last re-entry point (after the last successful checkpoint). A Psuedo Modified Restart occurs when the user requests additional output from the program which simply requires the re-execution of an output module such as the Structure Plotter, the Grid Point

Weight Generator, or the Stress Data Recovery module etc. The numerical solution is not affected by these modules; only output is generated. Note that a PMR is the common case since printer output, plotter output, etc. is usually requested. A true UMR is rare. On a PMR, output modules are re-executed to display requested output, and then the problem is continued at the re-entry point. A true Modified Restart occurs when some numerically significant data change. The modules which process this type of data must be re-executed. These modules are re-executed to regenerate their output data blocks based on the new data, and the problem is continued at the re-entry point. A Rigid Format Switch is a special form of Modified Restart in which a problem changes from one solution type to another. One example would be: a user has solved for the static solution on Rigid Format 1 and now wants to find the normal modes by using Rigid Format 3. This may or may not require data changes. The key difference here is that the re-entry point cannot be used to determine the proper place to restart. The technique by which a RFS is accomplished is to re-execute the final modules on the new Rigid Format and let the File Name Table chain the solution back to the proper restart point.

To understand, in general, how the above types of restart are implemented, it is necessary to consider the Module Execution Decision Table (MEDT), which is associated with each Rigid Format. The Module Execution Decision Table is a table which has one entry for every DMAP instruction in the Rigid Format. Each entry is 5 words long; each word contains 31 bits. For convenience, these bits are numbered sequentially from left to right with the numbers 1 through 155. If the entry in the MEDT for a module has, say, bit 55 turned on, this module will be executed whenever a card or data block change associated with bit 55 occurs on a restart. The Card Name Table associates bits of the MEDT with selected bulk data card names, Case Control selections and parameter names. The File Name Table associates bits of the MEDT with selected data block names. For consistency, bits 1 through 62 for each entry in the MEDT are reserved for the Card Name Table, and bits 94 through 155 are reserved for the File Name Table. The following example illustrates the use of these tables in determining the effects of changing a bulk data card on a Modified Restart. Suppose the FØRCE bulk data card is to be changed when executing Rigid Format 1. The table in section 3.2.3.1 of the User's Manual associates bit 60 with the FØRCE card. The decision table for bits 1 through 62 is shown in section 3.2.3.3 of the User's Manual. DMAP modules BEGIN, FILE, FILE, GP3, SAVE, PARAM, PURGE, CHKPNT, SSG1, CHKPNT, EQUIV, etc., will be executed since bit 60 is on for each.

There is one more table, the Rigid Format Switch Table, which is constant for all Rigid Formats, and hence resides in Preface module XCSA. The Rigid Format Switch Table associates with

each Rigid Format a bit for each entry in the MEDT: bit 63 is associated with Rigid Format 1, bit 64 is associated with Rigid Format 2, etc. If the restart involves a Rigid Format change, the bit in the decision table which is set is the bit corresponding to the Rigid Format of the previous execution.

Each part of the NASTRAN Preface contributes to processing the information for a restart. XCSA extracts and stores the Card Name Table, the File Name Table, the Module Execution Decision Table, the DMAP sequence and the Rigid Format Switch bit if any. These are written in the XCSA Executive Control Table (see section 2.4.2.5) on the New Problem Tape for later use by module XGPI. IFP1 compares the current CASECC data block with the one submitted on the previous run (a copy of CASECC is stored on the Old Problem Tape for this purpose). Three types of changes are noted by IFP1:

1. Changes in data set selection such as Load set, SPC set, etc.;
2. The occurence of output requests for printer, plotter, etc.;
3. Changes in the looping structure of the problem.

The results of this analysis are stored in common block /IFPXO/. Each bit in /IFPXO/ is associated with a key name. These names will appear in some Rigid Format's Card Name Table. /IFPXO/ contains one bit for every unique entry in the Card Name Table. /IFPX1/ contains these names in order given by /IFPXO/. Thus, bit 135 in /IFPXO/ corresponds to the key word, LØAD$. If bit 135 is on (non-zero), the status of LØAD$ has changed on this restart. XSØRT analyzes the bulk data card changes in a similar manner, setting the proper bits in /IFPXO/. IFP applies certain logical rules to combinations of the bits. XGPI then analyzes this information in the following manner. For each bit in /IFPXO/ the BCD equivalent is extracted from /IFPX1/. This mnemonic is searched for a match in the Card Name Table. If a match occurs, the appropriate bit in a master module execution mask is turned on. After all changes have been processed, the master mask is logically multiplied (logical and) with each module execution entry. A non-zero results indicates that this module is to be executed.

All bits in /IFPXO/ are classified as either significant to the solution or as only reflecting output requests. If only output request bits are on, a PMR is indicated. If the restart is a Modified Restart, one further table look-up may be necessary: the resulting DMAP sequence determined from the execution flags of the modules may not have the required input data blocks. (All

input data blocks must first appear as output data blocks). If this should be the case (most often caused by switching rigid formats), the File Name Table is consulted to determine which bits are on and hence which modules should be re-executed to generate the missing data blocks. The resulting DMAP sequence causes the selective execution of only those modules necessary to reflect the data changes and complete the requested solution.

2.1 INTRODUCTION

This section contains descriptions of a) those NASTRAN data blocks which appear in one or more Rigid Formats (section 2.3), b) Executive tables maintained by the NASTRAN Executive System (section 2.4), and c) Miscellaneous tables used by more than one module (section 2.5).

Data blocks that appear in Rigid Formats are structural problem oriented and reside on physical files. A file is "allocated" to a data block, and a data block is "assigned" to a file. The Executive Segment File Allocator (XSFA) Module is the "administrative manager" of files for NASTRAN.

Executive Tables have true executive functions in the sense that they are not oriented to a particular problem solution or even to structural analysis in general. They may be core resident or may reside on physical files.

Miscellaneous tables are common blocks which are used by the Executive System and/or a particular class of modules (e.g., /GPTA1/ is used only by modules GP1, GP2, GP3 and TA1). Common blocks that are used for intra-module communications are documented in section 4, Module Functional Descriptions.

Section 2.2.1 contains an index for data block descriptions sorted on data block names, and section 2.2.2 contains an index for data block descriptions sorted on the names of modules from which they are output. Alphabetical indexes are given for Executive table descriptions and miscellaneous table descriptions at the beginning of sections 2.4 and 2.5 respectively.

2.2 DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

Data block descriptions have been organized so that all data blocks output from the same module are grouped together. The name of each data block is given, and the data block is classified as a matrix or a table. A data block is classified as a matrix only if it is in NASTRAN matrix form, that is, generated by one of the NASTRAN matrix packing routines, PACK or BLDPK, the latter having secondary entry points BLDPKI, ZBLPKI and BLDPKN. All other data blocks are classified as tables.

Following a data block's name and classification is a brief description of its contents, followed by its format if it is a table. Since all matrices are in standard NASTRAN packed format a repeated description of the format is unnecessary for matrices. Each data block has a header record (consisting in general of two BCD words) which is the alphanumeric name of the data block as it appears in a Rigid Format, and this header record is designated "Record 0" in table formats. For those few data block which contain more than these two BCD words in their header record, e.g., SLT, GPTT, DLT, the contents are described explicitly. The conventions used for describing the types of words in records of tables are: R implies real; I implies integer; B implies BCD, four characters per computer word left adjusted with the remaining characters, if any, filled with BCD blanks; and L implies a "logical" -- not in the FØRTRAN sense -- word which is a mask of bits, right adjusted.

There is associated with each data block a six word control block called a trailer. Trailer information is "written" by the module which outputs the corresponding data block and can be "read" by any module accessing the corresponding data block as input. If a module "writes" a zero trailer for a data block, this implies no data was written in the data block. If a module "writes" a non-zero trailer, this implies data was written in the data block. Non-zero trailer information is often used by modules to allocate core storage before reading the corresponding data block. Trailer information for each data block is stored in and retrieved from the FIAT Executive table (see section 2.4.1.2) by the utility routines WRTTRL (write trailer) and RDTRL (read trailer), which are described in section 3.4.16. While residing in the FIAT, a trailer is stored in 6 half-words; each half-word consists of 16 binary digits.

Trailer information is standardized for matrix data blocks, not standardized for table data blocks. The format of a matrix trailer is as follows:

Word 1 = number of columns = N

Word 2 = number of rows = M

Word 3 = form = 1 square matrix

2 rectangular matrix

3 diagonal matrix { N = 1, M = number of rows }

4 lower triangular matrix

5 upper triangular matrix

6 symmetric matrix

7 row vector { N = 1, M = number of rows }

8 identity matrix

Word 4 = type = 1 elements of the matrix are real single precision

2 elements of the matrix are real double precision

3 elements of the matrix are complex single precision

4 elements of the matrix are complex double precision.

Word 5 = maximum number of non-zero words (rather than non-zero matrix elements) in any one column (e.g., if a real double precision matrix is diagonal and non-zero word 5 = 2)

Word 6 = not defined

Word 5 is dependent upon the structural model and the user's grid point sequencing rather than on any intrinsic property of the matrix and is therefore not described in this report.

The lower case letters, e.g., g, n, m, s, o, ℓ, etc., used as subscripts designate the subsets of displacement to which the root symbol (e.g., [K], for a stiffness matrix) applies. The reader is referred to section 3 of the Theoretical Manual and to section 1.7 of the Programmer's Manual for further details.

2.2.1 Index for Data Block Descriptions Sorted on Data Block Names

Section Number	Data Block Name	Output From Module	Page Number
2.3.47.2	AUTØ	RANDØM	2.3-180
2.3.2.11	AXIC	IFP	2.3-29
2.3.18.9	BAA	SMP1	2.3-72
2.3.41.2	BDD	GKAD	2.3-143
2.3.17.8	BFF	SCE1	2.3-68
2.3.10.2	BGG	SMA2	2.3-58
2.3.3.5	BGPDT	GP1	2.3-34
2.3.49.2	BHH	GKAM	2.3-183
2.3.16.4	BNN	MCE2	2.3-66
2.3.27.7	BQG	SDR1	2.3-85
2.3.41.8	B2DD	GKAD	2.3-145
2.3.40.3	B2PP	MTRXIN	2.3-142
2.3.1.1	CASECC	IFP1	2.3-1
2.3.39.1	CASEXX	CASE	2.3-141
2.3.42.2	CLAMA	CEAD	2.3-146
2.3.50.1	CPHID	DDR	2.3-184
2.3.27.11	CPHIP	SDR1	2.3-86
2.3.3.4	CSTM	GP1	2.3-33
2.3.27.9	DELTAPG	SDR1	2.3-86
2.3.27.10	DELTAQG	SDR1	2.3-86
2.3.27.8	DELTAUGV	SDR1	2.3-85
2.3.2.7	DIT	IFP	2.3-21
2.3.29.7	DLT	DPD	2.3-119
2.3.21.1	DM	RBMG3	2.3-77
2.3.2.9	DYNAMICS	IFP	2.3-25
2.3.8.3	ECPT	TA1	2.3-53
2.3.34.4	ECPTNL	PLA1	2.3-135
2.3.38.2	ECPTNL1	PLA4	2.3-140
2.3.4.1	ECT	GP2	2.3-36

Section Number	Data Block Name	Output From Module	Page Number
2.3.2.8	EDT	IFP	2.3-24
2.3.29.4	EED	DPD	2.3-116
2.3.5.4	ELSETS	PLTSET	2.3-38
2.3.2.5	EPT	IFP	2.3-16
2.3.29.5	EQDYN	DPD	2.3-118
2.3.3.2	EQEXIN	GP1	2.3-31
2.3.8.1	EST	TA1	2.3-45
2.3.34.2	ESTL	PLA1	2.3-132
2.3.34.3	ESTNL	PLA1	2.3-133
2.3.37.2	ESTNL1	PLA3	2.3-139
2.3.2.12	FØRCE	IFP	2.3-30
2.3.29.9	FRL	DPD	2.3-122
2.3.8.2	GEI	TA1	2.3-53
2.3.2.1	GEØM1	IFP	2.3-7
2.3.2.2	GEØM2	IFP	2.3-8
2.3.2.3	GEØM3	IFP	2.3-13
2.3.2.4	GEØM4	IFP	2.3-15
2.3.15.1	GM	MCE1	2.3-64
2.3.41.4	GMD	GKAD	2.3-144
2.3.18.1	GØ	SMP1	2.3-70
2.3.41.5	GØD	GKAD	2.3-144
2.3.8.4	GPCT	TA1	2.3-54
2.3.3.3	GPDT	GP1	2.3-32
2.3.3.1	GPL	GP1	2.3-31
2.3.29.1	GPLD	DPD	2.3-114
2.3.5.3	GPSETS	PLTSET	2.3-37
2.3.9.3	GPST	SMA1	2.3-56
2.3.7.2	GPTT	GP3	2.3-44
2.3.18.2	KAA	SMP1	2.3-70
2.3.33.2	KBFS	DSMG2	2.3-130
2.3.33.1	KBLL	DSMG2	2.3-130
2.3.33.3	KBSS	DSMG2	2.3-130

Section Number	Data Block Name	Output From Module	Page Number
2.3.32.1	KDAA	SMP2	2.3-129
2.3.35.3	KDAAM	ADD	2.3-137
2.3.41.1	KDD	GKAD	2.3-143
2.3.17.5	KDFF	SCE1	2.3-68
2.3.17.6	KDFS	SCE1	2.3-68
2.3.31.1	KDGG	DSMG1	2.3-128
2.3.16.3	KDNN	MCE2	2.3-65
2.3.17.7	KDSS	SCE1	2.3-68
2.3.17.1	KFF	SCE1	2.3-67
2.3.17.2	KFS	SCE1	2.3-67
2.3.12.1	KGG	SMA3	2.3-60
2.3.12.2	KGGL	SMA3	2.3-60
2.3.38.1	KGGNL	PLA4	2.3-140
2.3.35.1	KGGSUM	ADD	2.3-137
2.3.9.1	KGGX	SMA1	2.3-56
2.3.34.1	KGGXL	PLA1	2.3-132
2.3.49.3	KHH	GKAM	2.3-183
2.3.19.1	KLL	RBMG1	2.3-73
2.3.19.2	KLR	RBMG1	2.3-73
2.3.16.1	KNN	MCE2	2.3-65
2.3.18.3	KØØB	SMP1	2.3-70
2.3.19.3	KRR	RBMG1	2.3-73
2.3.17.3	KSS	SCE1	2.3-67
2.3.41.6	K2DD	GKAD	2.3-144
2.3.40.1	K2PP	MTRXIN	2.3-142
2.3.18.10	K4AA	SMP1	2.3-72
2.3.17.9	K4FF	SCE1	2.3-69
2.3.9.2	K4GG	SMA1	2.3-56
2.3.16.5	K4NN	MCE2	2.3-66
2.3.30.1	LAMA	READ	2.3-125
2.3.20.3	LBLL	RBMG2	2.3-75
2.3.20.1	LLL	RBMG2	2.3-75

Section Number	Data Block Name	Output From Module	Page Number
2.3.18.4	LØØ	SMP1	2.3-70
2.3.18.6	MAA	SMP1	2.3-71
2.3.2.10	MATPØØL	IFP	2.3-28
2.3.41.3	MDD	GKAD	2.3-143
2.3.5.5	MESSAGE	PLØT	2.3-39
2.3.17.4	MFF	SCE1	2.3-67
2.3.10.1	MGG	SMA2	2.3-58
2.3.49.1	MHH	GKAM	2.3-183
2.3.30.3	MI	READ	2.3-126
2.3.19.4	MLL	RBMG1	2.3-73
2.3.19.5	MLR	RBMG1	2.3-74
2.3.16.2	MNN	MCE2	2.3-65
2.3.18.8	MØAB	SMP1	2.3-72
2.3.18.7	MØØB	SMP1	2.3-71
2.3.2.6	MPT	IFP	2.3-20
2.3.22.1	MR	RBMG4	2.3-78
2.3.19.6	MRR	RBMG1	2.3-74
2.3.41.7	M2DD	GKAD	2.3-144
2.3.40.2	M2PP	MTRXIN	2.3-142
2.3.29.10	NLFT	DPD	2.3-122
2.3.28.21	ØBEF1	SDR2	2.3-108
2.3.28.17	ØBES1	SDR2	2.3-104
2.3.28.10	ØBQG1	SDR2	2.3-97
2.3.42.3	ØCEIGS	CEAD	2.3-147
2.3.28.14	ØCPHIP	SDR2	2.3-101
2.3.28.20	ØEFB1	SDR2	2.3-107
2.3.28.22	ØEFC1	SDR2	2.3-109
2.3.45.13	ØEFC2	SDR3	2.3-172
2.3.28.19	ØEF1	SDR2	2.3-106
2.3.45.5	ØEF2	SDR3	2.3-164
2.3.30.4	ØEIGS	READ	2.3-126

Section Number	Data Block Names	Output From Module	Page Number
2.3.28.16	ØESB1	SDR2	2.3-103
2.3.28.18	ØESC1	SDR2	2.3-105
2.3.45.12	ØESC2	SDR3	2.3-171
2.3.28.15	ØES1	SDR2	2.3-102
2.3.45.4	ØES2	SDR3	2.3-163
2.3.14.1	ØGPST	GPSP	2.3-63
2.3.11.1	ØGPWG	GPWG	2.3-59
2.3.37.1	ØNLES	PLA3	2.3-139
2.3.28.5	ØPG1	SDR2	2.3-92
2.3.43.1	ØPHID	VDR	2.3-149
2.3.28.13	ØPHIG	SDR2	2.3-100
2.3.43.5	ØPHIH	VDR	2.3-153
2.3.43.4	ØPNL1	VDR	2.3-152
2.3.45.6	ØPNL2	SDR3	2.3-154
2.3.28.7	ØPPC1	SDR2	2.3-94
2.3.45.9	ØPPC2	SDR3	2.3-168
2.3.28.6	ØPP1	SDR2	2.3-93
2.3.45.1	ØPP2	SDR3	2.3-160
2.3.28.9	ØQBG1	SDR2	2.3-96
2.3.28.8	ØQG1	SDR2	2.3-95
2.3.28.12	ØQPC1	SDR2	2.3-99
2.3.45.10	ØQPC2	SDR3	2.3-169
2.3.28.11	ØQP1	SDR2	2.3-98
2.3.45.2	ØQP2	SDR3	2.3-161
2.3.28.2	ØUBGV1	SDR1	2.3-89
2.3.43.2	ØUDVC1	VDR	2.3-150
2.3.45.14	ØUDVC2	SDR3	2.3-173
2.3.43.3	ØUDV1	VDR	2.3-151
2.3.45.7	ØUDV2	SDR3	2.3-166
2.3.28.1	ØUGV1	SDR2	2.3-88
2.3.43.6	ØUHVC1	VDR	2.3-154

Section Number	Data Block Name	Output From Module	Page Number
2.3.45.15	ØUHVC2	SDR3	2.3-174
2.3.43.7	ØUHV1	VDR	2.3-156
2.3.45.8	ØUHV2	SDR3	2.3-167
2.3.28.4	ØUPVC1	SDR2	2.3-91
2.3.45.11	ØUPVC2	SDR3	2.3-170
2.3.28.3	ØUPV1	SDR2	2.3-90
2.3.45.3	ØUPV2	SDR3	2.3-162
2.3.53.2	PAF	DDR2	2.3-192
2.3.53.5	PAT	DDR2	2.3-193
2.3.33.4	PBL	DSMG2	2.3-131
2.3.33.5	PBS	DSMG2	2.3-131
2.3.1.2	PCDB	IFP1	2.3-3
2.3.44.3	PDF	FRRD	2.3-158
2.3.48.2	PDT	TRD	2.3-181
2.3.23.1 , 2.3.35.2	PG	SSG1 , ADD	2.3-79, 2.3-137
2.3.27.2	PGG	SDP1	2.3-84
2.3.23.2	PG1	SSG1	2.3-79
2.3.36.2	PGV1	PLA2	2.3-138
2.3.30.2	PHIA	READ	2.3-125
2.3.42.1	PHID	CEAD	2.3-146
2.3.49.4	PHIDH	GKAM	2.3-183
2.3.27.4	PHIG	SDR1	2.3-84
2.3.42.4	PHIH	CEAD	2.3-148
2.3.24.4	PL	SSG2	2.3-80
2.3.26.1	PLI	SSG4	2.3-83
2.3.6.1	PLØTX1	PLØT	2.3-40
2.3.6.2	PLØTX2	PLØT	2.3-40
2.3.5.2	PLTPAR	PLTSET	2.3-37
2.3.5.1	PLTSETX	PLTSET	2.3-37
2.3.48.5	PNLD	TRD	2.3-182
2.3.48.7	PNLH	TRD	2.3-182
2.3.24.2	PØ	SSG2	2.3-80

Section Number	Data Block Name	Output From Module	Page Number
2.3.26.2	PØI	SSG4	2.3-83
2.3.44.4	PPF	FRRD	2.3-158
2.3.28.25	PPHIG	SDR2	2.3-112
2.3.48.4	PPT	TRD	2.3-182
2.3.24.3	PS	SSG2	2.3-80
2.3.47.1	PSDF	RANDØM	2.3-179
2.3.29.8	PSDL	DPD	2.3-121
2.3.44.2	PSF	FRRD	2.3-158
2.3.48.3	PST	TRD	2.3-181
2.3.28.24	PUBGV1	SDR2	2.3-111
2.3.28.26	PUGV	SDR2	2.3-113
2.3.28.23	PUGV1	SDR2	2.3-110
2.3.27.6	QBG	SDR1	2.3-85
2.3.27.3	QG	SDR1	2.3-84
2.3.36.3	QG1	PLA2	2.3-138
2.3.27.15	QP	SDR1	2.3-87
2.3.27.12	QPC	SDR1	2.3-86
2.3.24.1	QR	SSG2	2.3-80
2.3.13.1	RG	GP4	2.3-61
2.3.25.6	RUBLV	SSG3	2.3-82
2.3.25.3	RULV	SSG3	2.3-81
2.3.25.4	RUØV	SSG3	2.3-81
2.3.3.6	SIL	GP1	2.3-35
2.3.29.2	SILD	DPD	2.3-114
2.3.7.1	SLT	GP3	2.3-41
2.3.29.6	TFPØØL	DPD	2.3-119
2.3.29.11	TRL	DPD	2.3-124
2.3.27.5	UBGV	SDR1	2.3-85
2.3.20.4	UBLL	RBMG2	2.3-76
2.3.25.5	UBLV	SSG3	2.3-82
2.3.33.7	UBØØV	DSMG2	2.3-131
2.3.53.4	UEVT	DDR2	2.3-192

Section Number	Data Block Name	Output From Module	Page Number
2.3.44.1	UDVF	FRRD	2.3-158
2.3.50.2	UDV1F	DDR1	2.3-184
2.3.53.3	UDV2F	DDR2	2.3-192
2.3.48.1	UDVT	TRD	2.3-181
2.3.50.3	UDV1T	DDR1	2.3-184
2.3.53.6	UDV2T	DDR2	2.3-193
2.3.53.1	UEVF	DDR2	2.3-192
2.3.27.1	UGV	SDR1	2.3-84
2.3.36.1	UGV1	PLA2	2.3-138
2.3.44.5	UHVF	FRRD	2.3-159
2.3.48.6	UHVT	TRD	2.3-182
2.3.20.2	ULL	RBMG2	2.3-75
2.3.25.1	ULV	SSG3	2.3-81
2.3.27.14	UPV	SDR1	2.3-87
2.3.27.13	UPVC	SDR1	2.3-87
2.3.18.5	UØØ	SMP1	2.3-71
2.3.25.2	UØØV	SSG3	2.3-81
2.3.13.3	USET	GP4	2.3-61
2.3.29.3	USETD	DPD	2.3-115
2.3.1.3	XYCDB	IFP1	2.3-4
2.3.46.2	XYPLTFA	XYTRAN	2.3-178
2.3.46.3	XYPLTF	XYTRAN	2.3-178
2.3.46.4	XYPLTR	XYTRAN	2.3-178
2.3.46.1	XYPLTT	XYTRAN	2.3-175
2.3.46.5	XYPLTTA	XYTRAN	2.3-178
2.3.33.6	YBS	DSMG2	2.3-131
2.3.13.2	YS	GP4	2.3-61

2.2.2 Index for Data Block Descriptions Sorted Alphabetically by Module

Section Number	Module	Page Number	Section Number	Module	Page Number
2.3.35	ADD	2.3-137	2.3.38	PLA4	2.3-140
2.3.39	CASE	2.3-141	2.3.6	PLØT	2.3-40
2.3.42	CEAD	2.3-146	2.3.5	PLTSET	2.3-37
2.3.50	DDR1	2.3-184	2.3.47	RANDØM	2.3-179
2.3.53	DDR2	2.3-192	2.3.19	RBMG1	2.3-73
2.3.29	DPD	2.3-114	2.3.20	RBMG2	2.3-75
2.3.31	DSMG1	2.3-128	2.3.21	RBMG3	2.3-77
2.3.33	DSMG2	2.3-130	2.3.22	RBMG4	2.3-78
2.3.44	FRRD	2.3-158	2.3.30	READ	2.3-125
2.3.41	GKAD	2.3-143	2.3.17	SCE1	2.3-67
2.3.49	GKAM	2.3-183	2.3.27	SDR1	2.3-84
2.3.3	GP1	2.3-31	2.3.28	SDR2	2.3-88
2.3.4	GP2	2.3-36	2.3.45	SDR3	2.3-160
2.3.7	GP3	2.3-41	2.3.9	SMA1	2.3-56
2.3.13	GP4	2.3-61	2.3.10	SMA2	2.3-58
2.3.14	GPSP	2.3-63	2.3.12	SMA3	2.3-60
2.3.11	GPWG	2.3-59	2.3.18	SMP1	2.3-70
2.3.2	IFP	2.3-5	2.3.32	SMP2	2.3-129
2.3.1	IFP1	2.3-1	2.3.23	SSG1	2.3-79
2.3.15	MCE1	2.3-64	2.3.24	SSG2	2.3-80
2.3.16	MCE2	2.3-65	2.3.25	SSG3	2.3-81
2.3.40	MTRXIN	2.3-142	2.3.26	SSG4	2.3-83
2.3.34	PLA1	2.3-132	2.3.8	TA1	2.3-45
2.3.36	PLA2	2.3-138	2.3.48	TRD	2.3-181
2.3.37	PLA3	2.3-139	2.3.43	VDR	2.3-149
			2.3.46	XYTRAN	2.3-175

2.3 DATA BLØCK DESCRIPTIONS

2.3.1 Data Blocks Output From Module IFP1

2.3.1.1 CASECC (TABLE)

Description

Case Control Data Table

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Subcase number
	2	I	Multipoint constraint set
	3	I	Single-point constraint set
	4	I	External static load set
	5	I	Real eigenvalue extraction set
	6	I	Element deformation set
	7	I	Thermal load set
	8	I	Thermal material set
	9	I	Transient initial conditions
	10	I	Non-linear load output set
	11	I	Output media selection - 1 = print, 4 = punch
	12	I	Format of output - 1 = real 2 = real/imag 3 = mag/phase If word 12 < 0, SØRT2 is requested
	13	I	Dynamic load set
	14	I	Frequency response set
	15	I	Transfer function set
	16	I	Symmetry flag
	17	I	
	18	I	Same as 10-12 for load output
	19	I	
	20	I	
	21	I	Same as 10-12 for displacement output
	22	I	
	23	I	
	24	I	Same as 10-12 for stress output
	25	I	
	26	I	
	27	I	Same as 10-12 for force output
	28	I	
	29	I	
	30	I	Same as 10-12 for acceleration output
	31	I	
	32	I	
	33	I	Same as 10-12 for velocity output
	34	I	
	35	I	
	36	I	Same as 10-12 for forces of constraint output
	37	I	
	38	I	Time step set selection for transient problem
	39	B	
	.		
	.	B	32 words of TITLE
	.		
	70	B	

Record	Word	Type	Item
	71	B	32 words of SUBTITLE
	.		
	.	B	
	.		
	102	B	
	103	B	32 words of LABEL
	.		
	.	B	
	.		
	134	B	
	135	I	Structure plotter flag
	136	I	Axisymmetric set - 2 = COSINE 1 = SINE
	137	I	Number of harmonics to output
	138	I	Differential stiffness coefficient set
	139	B	Name of K2PP matrix
	140	B	
	141	B	Name of M2PP matrix
	142	B	
	143	B	Name of B2PP matrix
	144	B	
	145	I	OUTPUT frequency set selection
	146		Not defined
	147		Not defined
	148	I	Complex eigenvalue extraction set
	149	I	Structural damping table set
	150	I	Inertia relief set (Force method only)
	151	I	Same as 10-12 for solution set displacements
	152	I	
	153	I	
	154	I	Same as 10-12 for solution set velocities
	155	I	
	156	I	
	157	I	Same as 10-12 for solution set accelerations
	158	I	
	159	I	
	160	I	Non-linear load set in transient problems
	161	I	Delete set (Force method only)
	162		Not defined
	163	I	Random analysis set
	164	I	Piecewise linear coefficient set
	165		Not defined
	166	I	Length of symmetry sequence (LSEM)
	167	R	Coefficients for symmetry sequence
	.		
	.	R	
	.		
	166+LSEM	R	
	167+LSEM	I	Set ID
	168+LSEM	I	Length of the set (LSET)
	169+LSEM		Set members
	.		
	.		
	.		
	169+LSEM+LSET	I	

Note

The above record is repeated for each subcase and symmetry combination.

Table Trailer

Word 1 = 0
Word 2 = 7
Word 3 = 0
Word 4 = 7
Word 5 = 0
Word 6 = 7

2.3.1.2 PCDB (TABLE)

Description

Plot Control Data Table for the structure plotter.

Table Format

Record	Item
0	Header record
1	The data here is the XRCARD translation of the Structure Plotter
.	Packet cards in the Case Control Deck (See Subroutine Description
.	for XRCARD). There is one record for each physical card.
.	
N	
N+1	End-of-file

Table Trailer

Words 1 through 3 are zero
Word 4 = 7777
Word 5 and Word 6 are zero

2.3.1.3 XYCDB (TABLE)

Description

XY Output Control Data Block.

Record one contains the subroutine IFP1XY interpretations of the XY plot output request case control data cards. Record two contains an N by 6 matrix stored by rows and sorted such that the columns are in sort left to right. N is the total number of combinations specified by the XY-plot-request case control data cards.

Table Format

Record	Word	Type	Item	
0			Header record	
1	1-Last	Mixed	IFP1XY interpretations of the XY-output-requests, translated for rapid processing by the XYTRAN module.	
2	1	I	Subcase number (0 implies all)	repeats for all rows
	2	I	Vector request type	
	3	I	Point or element ID	
	4	I	Component	
	5	I	XY output type	
	6	I	Destination code	

Notes

1. Vector request type =
   1 = Displacement
   2 = Velocity
   3 = Acceleration
   4 = Single-point forces of constraint
   5 = Load
   6 = Stress
   7 = Force
   8 = Adisplacement
   9 = Avelocity
   10 = Aacceleration
   11 = Nonlinear Force

2. XY output type =
   1 = Response
   2 = PSDF
   3 = AUTØ

3. Destination code =
   1 = Print
   2 = Plot
   3 = Print, plot
   4 = Punch
   5 = Print, punch
   6 = Plot, punch
   7 = Print, plot, punch

4. Either of records 1 and 2 may be null, however they will always exist.

Table Trailer

Words 1-5 = nonzero
Word 6 = one

2.3.2 Data Blocks Output From Module IFP.

Module IFP (Input File Processor, part of the NASTRAN Preface) processes the Bulk Data Deck sorted by module XSØRT and writes the following data blocks that appear in one or more Rigid Formats: GEØM1, GEØM2, GEØM3, GEØM4, EPT, MPT, DIT, EDT, DYNAMICS, MATPØØL. Each one of these data blocks has the usual 2-word BCD header record.

Each of these data blocks contains bulk data card images or modified card images of a subset of bulk data card types. Each logical record of each of the above data blocks contains all the data of a particular card type. If a card type is not present in the Bulk Data Deck, there is no record. For each card type present, 3 words are written as header information for the record. Then for every logical bulk data card of that type in the Bulk Data Deck, a card image or a modified card image is written sequentially in the record.

The first two words of the header information of each record are used by entry point LØCATE of subroutine PRELØC. The third word of the header information is the card number used by the IFP programmer to reference tables internal to the IFP module.

LØCATE is used by routines that wish to read data blocks output by the IFP. The second word of the header information portion of each record corresponds to a bit position in a 96-bit, 6-word data block trailer, each word containing 16 bits. If a routine requests LØCATE to locate (position the file to) a particular card type, LØCATE will determine if the card type is present by interrogating the corresponding bit in the trailer, the bit number having been supplied through the calling sequence to LØCATE. If the bit is zero, the card type is not present and LØCATE executes a non-standard RETURN. If the bit is one, the card type is present and LØCATE uses the first word of the header information, also supplied through the calling sequence, to find the proper logical record (card type).

Since the Bulk Data Deck is sorted alphabetically before IFP processes it, and since IFP processes the deck sequentially, the order of the card types in each data block is alphabetical. It should be noted that when two trailer bit numbers are given in the description for a card type, this implies that: (1) the card is a "multi-entry" card type (more than one logical card on one physical card, e.g., CELAS3, CELAS4, CRØD, CTUBE); (2) these card types may or may not be sorted with respect to the primary field, in the above examples element identification; and (3) a module accessing these card types must know whether or not they are in sort. If they are in sort, the second bit number is set to 0; if they are not in sort, the second bit number is set to 1.

Card type formats correspond to a typical card entry in the logical record allocated to a card type. A number (literal) in a card type format implies that the IFP has placed this number in its output buffer before writing the information on the file and that this number is not on the bulk data card itself. The mnemonics used in the card type formats correspond to the mnemonics in the bulk data card section of the NASTRAN User's Manual. It is advised that anyone using the information on the following pages secure a copy of this section of the User's Manual for cross reference purposes.

IFP also generates the AXIC and FØRCE data blocks, the Parameter Value Table (PVT) and writes Direct Matrix Input (DMI) and Direct Table Input (DTI) cards on the Data Pool File. The AXIC data block, whose presence implies a conical shell (a unique structural element) problem solution, is processed by the Preface module IFP3. The FØRCE data block is used in the force method only. The PVT, which is an Executive Table and is documented in section 2.4, contains the names and values of all parameters input by means of the PARAM bulk data card. The PVT is written on the Problem Tape. Each DMI in the Bulk Data Deck is output on the Data Pool File in NASTRAN packed matrix format and is indistinguishable from any matrix data block pooled by the XSFA, that is, a matrix trailer is written as the last logical record of the data block. IFP also stores the name of each DMI in the DPL (see section 2.4). Similarly, each DTI is output on the Data Pool File, a table trailer is written, and the name of the DTI is stored in the DPL.

2.3.2.1 GEØM1 (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
CØRD1C	1701	17	6
CØRD1R	1801	18	5
CØRD1S	1901	19	7
CØRD2C	2001	20	9
CØRD2R	2101	21	8
CØRD2S	2201	22	10
GRID	4501	45	1
SEQGP	5301	53	4

Card Type Formats:

CØRD1C (6 words)	CID	2	1
	G1	G2	G3
CØRD1R (6 words)	CID	1	1
	G1	G2	G3
CØRD1S (6 words)	CID	3	1
	G1	G2	G3
CØRD2C (13 words)	CID	2	2
	RID	A1	A2
	A3	B1	B2
	B3	C1	C2
	C3		
CØRD2R (13 words)	CID	1	2
	RID	A1	A2
	A3	B1	B2
	B3	C1	C2
	C3		
CØRD2S (13 words)	CID	3	2
	RID	A1	A2
	A3	B1	B2
	B3	C1	C2
	C3		
GRID (8 words)	ID	CP	X1
	X2	X3	CD
	PS	FØ	

If a GRDSET card is present, and if any of fields 3, 7, 8, 9 of any GRID card are blank, IFP will insert corresponding data fields from the GRDSET card. Only one GRDSET card may appear in the Bulk Data Deck.

SEQGP (2 words)	ID	SEQID

2.3.2.2 GEØM2 (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
CBAR	2408	24	180
CDAMP1	201	2	69
CDAMP2	301	3	70
CDAMP3	401	4,92	71
CDAMP4	501	5,89	72
CELAS1	601	6	73
CELAS2	701	7	74
CELAS3	801	8,91	75
CELAS4	901	9,88	76
CGENEL	2008	20	186
CMASS1	1001	10	65
CMASS2	1101	11	66
CMASS3	1201	12,93	67
CMASS4	1301	13,90	68
CØNM1	1401	14	63
CØNM2	1501	15	64
CØNRØD	1601	16	47
CQDMEM	2601	26	60
CQDPLT	2701	27	59
CQUAD1	2801	28	57
CQUAD2	2901	29	58
CQUAD3	2958	40	167
CRØD	3001	30,96	48
CSHEAR	3101	31	61
CTØRDRG	1908	19	104
CTRAPRG	1808	18	80
CTRBSC	3201	32	54
CTRIA1	3301	33	52
CTRIA2	3401	34	53
CTRIA3	3458	41	114
CTRIARG	1708	17	79
CTRMEM	3501	35	56
CTRPLT	3601	36	55
CTUBE	3701	37,95	49
CTWIST	3801	38	62
CVISC	3901	39,94	50
GENEL	4301	43	28
PLØTEL	5201	52,87	11
QDSEP	4808	48	108
REACT	5251	62	88
SECL	7308	73	176
SECP	7408	74	177
SEQD	7508	75	106
SEQDBFE	4608	46	107
SETR	7608	76	170
SETRBFE	4708	47	116
SPØINT	5551	49	105

Card Type Formats:

CBAR (16 words)	EID	PID	GA
	GB	X1	X2
	X3	F	PA
	PB	Z1A	Z2A
	Z3A	Z1B	Z2B
	Z3B		

If a BARØR card is present in the Bulk Data Deck, and if any of the fields 3, 6, 7, 8, 9 of any CBAR card are blank, IFP will insert the corresponding data fields from the BARØR card. Only one BARØR card may appear in the Bulk Data Deck.

CDAMP1 (6 words)	EID	PID	G1
	G2	C1	C2
CDAMP2 (6 words)	EID	B	G1
	G2	C1	C2
CDAMP3 (4 words)	EID	PID	S1
	S2		
CDAMP4 (4 words)	EID	B	S1
	S2		
CELAS1 (6 words)	EID	PID	G1
	G2	C1	C2
CELAS2 (8 words)	EID	K	G1
	G2	C1	C2
	GE	S	
CELAS3 (4 words)	EID	PID	S1
	S2		
CELAS4 (4 words)	EID	K	S1
	S2		
CGENEL	Not available.		
CMASS1 (6 words)	EID	PID	G1
	G2	C1	C2
CMASS2 (6 words)	EID	M	G1
	G2	C1	C2
CMASS3 (4 words)	EID	PID	S1
	S2		
CMASS4 (4 words)	EID	M	S1
	S2		

Card Type Formats Cont'd.:

CØNM1 (24 words)	EID	G	CID
	M11	M21	M22
	M31	M32	M33
	M41	M42	M43
	M44	M51	M52
	M53	M54	M55
	M61	M62	M63
	M64	M65	M66
CØNM2 (13 words)	EID	G	CID
	M	X1	X2
	X3	I11	I21
	I22	I31	I32
	I33		
CØNRØD (8 words)	EID	G1	G2
	MID	A	J
	C	NSM	
CQDMEM (7 words)	EID	PID	G1
	G2	G3	G4
	TH		
CQDPLT (7 words)	EID	PID	G1
	G2	G3	G4
	TH		
CQUAD1 (7 words)	EID	PID	G1
	G2	G3	G4
	TH		
CQUAD2 (7 words)	EID	PID	G1
	G2	G3	G4
	TH		
CQUAD3 (7 words)	EID	PID	G1
	G2	G3	G4
	TH		
CRØD (4 words)	EID	PID	G1
	G2		
CSHEAR (6 words)	EID	PID	G1
	G2	G3	G4
CTØRDRG (7 words)	EID	PID	G1
	G2	A1	A2
	O		
CTRAPRG (7 words)	EID	G1	G2
	G3	G4	TH
	MID		
CTRBSC (6 words)	EID	PID	G1
	G2	G3	TH
CTRIA1 (6 words)	EID	PID	G1
	G2	G3	TH
CTRIA2 (6 words)	EID	PID	G1
	G2	G3	TH

Card Type Formats Cont'd.:

CTRIA3 (6 words)	EID	PID	G1
	G2	G3	TH
CTRIARG (6 words)	EID	G1	G2
	G3	TH	MID
CTRMEM (6 words)	EID	PID	G1
	G2	G3	TH
CTRPLT (6 words)	EID	PID	G1
	G2	G3	TH
CTUBE (4 words)	EID	PID	G1
	G2		
CTWIST (6 words)	EID	PID	G1
	G2	G3	G4
CVISC (4 words)	EID	PID	G1
	G2		
GENEL (open ended)	EID	UI1	CI1
	UI2	CI2	...
	UIM	CIM	-1
	M	UD1	CD1
	UD2	CD2	...
	UDN	CDN	-1
	N	Z11	Z21
	Z22	Z31	Z32
	Z33	...	ZMM
	N	S11	S12
	...	S1N	S21
	S22	...	S2N
	...	SM1	SM2
	...	SMN	
PLØTEL (3 words)	EID	G1	G2
QDSEP (3 words)	SEID	G	SPID
REACT (19 words)	EID	G	FE
	LS	RID	VTY
	VID	Z11	Z22
	Z33	Z44	Z55
	Z66	SF1	SF2
	SF3	SM4	SM5
	SM6		
SECL	Not available.		
SECP	Not available.		
SEQD (open ended)	QDTYPE (2 words)	SEID	AB
	ABID	BFE	TVS
	FE	0	EID1
	EID2	...	EIDN
	-1		
SEQDBFE (3 words)	SEID	G	BFE

Card Type Formats Cont'd.:

SETR	Not available.
SETRBFE	Not available.
SPØINT (1 word)	ID

2.3.2.3 GEØM3 (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
FØRCE	4201	42	18
FØRCE1	4001	40	20
FØRCE2	4101	41	22
GRAV	4401	44	26
LØAD	4551	61	84
MØMENT	4801	48	19
MØMENT1	4601	46	21
MØMENT2	4701	47	23
PLØAD	5101	51	24
PLØAD2	6802	68	199
RFØRCE	5509	55	190
SLØAD	5401	54	25
TEMP	5701	57	27
TEMPD	5641	65	98

Card Type Formats:

FØRCE (7 words)	SID	G	CID
	F	N1	N2
	N3		
FØRCE1 (5 words)	SID	G	F
	G1	G2	
FØRCE2 (7 words)	SID	G	F
	G1	G2	G3
	G4		
GRAV (6 words)	SID	CID	G
	N1	N2	N3
LØAD (open ended)	SID	S	S1
	L1	S2	L2
	...	S_n	L_n
	-1	-1	
MØMENT (7 words)	SID	G	CID
	M	N1	N2
	N3		
MØMENT1 (5 words)	SID	G	M
	G1	G2	
MØMENT2 (7 words)	SID	G	M
	G1	G2	G3
	G4		
PLØAD (6 words)	SID	P	G1
	G2	G3	G4
PLØAD2 (3 words)	SID	P	EID
RFØRCE (7 words)	SID	G	CID
	A	N1	N2
	N3		

Card Type Formats Cont'd.:

SLØAD (3 words)	SID	S	F
TEMP (3 words)	SID	G	T
TEMPD (2 words)	SID	T	

2.3.2.4 GEØM4 (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
MPC	4901	49	17
MPCADD	4891	60	83
ØMIT	5001	50	15
ØMIT1	4951	63	92
SPC	5501	55	16
SPC1	5481	58	12
SPCADD	5491	59	13
SUPØRT	5601	56	14

Card Type Formats:

MPC (open ended)	SID	G	C
	A	G	C
	A	...	G
	C	A	-1
	-1	-1	
MPCADD (open ended)	SID	S1	S2
	...	S_n	-1
ØMIT (2 words)	ID	C	

Components can be input in any unique combination of digits 1-6. Output format will be ID and one digit, the digits for any one entry being in sort.

Example:	ID	C	
	12	3516	
Output as:	12	1	12
	3	12	5
	12	6	
ØMIT1 (open ended)	C	G	G
	...	G	-1
SPC (4 words)	SID	G	C
	D		
SPC1 (open ended)	SID	C	G1
	G2	...	G_n
	-1		
SPCADD (open ended)	SID	S1	S2
	...	S_n	-1
SUPØRT (2 words)	ID	C	

The note above concerning the ØMIT card applies to the SUPØRT card as well.

2.3.2.5 EPT (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
PBAR	52	20	181
PCØNEAX	152	19	147
PDAMP	202	2	45
PELAS	302	3	46
PGENEL	352	26	187
PMASS	402	4	44
PQDMEM	502	5	41
PQDPLT	602	6	40
PQUAD1	702	7	38
PQUAD2	802	8	39
PQUAD3	852	49	168
PRØD	902	9	29
PSHEAR	1002	10	42
PTØRDRG	2102	21	121
PTRBSC	1102	11	35
PTRIA1	1202	12	33
PTRIA2	1302	13	34
PTRIA3	1352	24	115
PTRMEM	1402	14	37
PTRPLT	1502	15	36
PTUBE	1602	16	30
PTWIST	1702	17	43
PVISC	1802	18	31
SPQDMEM	6302	63	111
SPQDPLT	6402	64	112
SPQUAD1	6502	65	109
SPQUAD2	6602	66	110
SPQUAD3	6702	67	169
SPTRBSC	7302	73	174
SPTRIA1	7402	74	171
SPTRIA2	7502	75	172
SPTRIA3	7602	76	178
SPTRMEM	7702	77	173
SPTRPLT	7802	78	175

Card Type Formats:

PBAR (19 words)	PID	MID	A
	I1	I2	J
	NSM	FE	C1
	C2	D1	D2
	E1	E2	F1
	F2	K1	K2
	I12		
PCØNEAX (24 words)	ID	MID1	T1
	MID2	I	MID3
	T2	NSM	Z1
	Z2	PHI1	PHI2
	PHI3	PHI4	PHI5
	PHI6	PHI7	PHI8
	PHI9	PHI10	PHI11
	PHI12	PHI13	PHI14
PDAMP (2 words)	PID	B	

Card Type Formats Cont'd.:

PELAS (4 words)	PID	K	GE
	S		
PGENEL	Not available.		
PMASS (2 words)	PID	M	
PQDMEM (4 words)	PID	MID	T
	NSM		
PQDPLT (8 words)	PID	MID1	I
	MID2	T	NSM
	Z1	Z2	
PQUAD1 (10 words)	PID	MID1	T1
	MID2	I	MID3
	T3	NSM	Z1
	Z2		
PQUAD2 (4 words)	PID	MID	T
	NSM		
PQUAD3 (62 words)	SPID	RA	TO
	ZAA	ZBB	ZAB
	HS	Z	A11
	A21	A31	A41
	A51	A61	A12
	A22	A32	A42
	A52	A62	Z11
	Z22	Z33	Z44
	Z55	Z66	Z34
	Z12	Z13	Z23
	Z45	Z46	Z56
	Z35	Z14	Z15
	Z16	Z24	Z25
	Z26	Z36	SU11
	SU14	SU22	SU25
	SU33	SU36	STA
	STB	SL11	SL14
	SL22	SL25	SL33
	SL36	SAT	SBT
	SAC	SBC	TAB
	TAN	TBN	
PRØD (6 words)	PID	MID	A
	J	C	NSM
PSHEAR (4 words)	PID	MID	T
	NSM		
PTØRDRG (4 words)	PID	MID	TM
	TF		
PTRBSC (8 words)	PID	MID1	I
	MID2	T	NSM
	Z1	Z2	

Card Type Formats Cont'd.:

PTRIA1 (10 words)	PID	MID1	T1
	MID2	I	MID3
	T3	NSM	Z1
	Z2		
PTRIA2 (4 words)	PID	MID	T
	NSM		
PTRIA3	Not available.		
PTRMEM (4 words)	PID	MID	T
	NSM		
PTRPLT (8 words)	PID	MID1	I
	MID2	T2	NSM
	Z1	Z2	
PTUBE (5 words)	PID	MID	ØD
	T	NSM	
PTWIST (4 words)	PID	MID	T
	NSM		
PVISC (3 words)	PID	C1	C2
SPQDMEM (4 words)	SPID	MID	H
	NSM		
SPQDPLT (8 words)	SPID	MID1	I
	MID2	HS	NSM
	Z1	Z2	
SPQUAD1 (10 words)	SPID	MID1	H1
	MID2	I	MID3
	HS	NSM	Z1
	Z2		
SPQUAD2 (4 words)	SPID	MID1	H
	NSM		
SPQUAD3 (62 words)	SPID	RA	TO
	ZAA	ZBB	ZAB
	HS	Z	A11
	A21	A31	A41
	A51	A61	A12
	A22	A32	A42
	A52	A62	Z11
	Z22	Z33	Z44
	Z55	Z66	Z34
	Z12	Z13	Z23
	Z45	Z46	Z56
	Z35	Z14	Z15
	Z16	Z24	Z25
	Z26	Z36	SU11
	SU14	SU22	SU25
	SU33	SU36	STA
	STB	SL11	SL14
	SL22	SL25	SL33
	SL36	SAT	SBT
	SAC	SBC	TAB
	TAN	TBN	

Card Type Formats Cont'd.:

SPTRBSC	Not available.
SPTRIA1	Not available.
SPTRIA2	Not available.
SPTRIA3	Not available.
SPTRMEM	Not available.
SPTRPLT	Not available.

2.3.2.6 MPT (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
DSFACT	53	10	143
MAT1	103	1	77
MAT2	203	2	78
MAT3	1403	14	122
MATS1	503	5	90
MATT1	703	7	91
MATT2	803	8	102
MATT3	1503	15	189
PLFACT	1103	11	185

Card Type Formats:

DSFACT (open ended)	SID	B1	B2
	...	B_n	-1
MAT1 (11 words)	MID	E	G
	NU	RHØ	A
	TREF	GE	ST
	SC	SS	

If any one of E, G or NU is blank, it will be computed to satisfy the identity E = 2 (1+NU)G; otherwise, values supplied by the user will be used.

MAT2 (16 words)	MID	G11	G12
	G13	G22	G23
	G33	RHØ	A1
	A2	A12	TO
	GE	ST	SC
	SS		
MAT3 (16 words)	MID	EX	EY
	EZ	NUXY	NUYZ
	NUZX	RHØ	GXY
	GYZ	GZX	AX
	AY	AZ	TREF
	GE		
MATS1 (11 words)	MID	R1	R2
	R3	R4	...
	R10		
MATT1 (11 words)	MID	R1	R2
	R3	R4	...
	R10		
MATT2 (16 words)	MID	R1	R2
	R3	R4	...
	R15		
MATT3 (16 words)	MID	R1	R2
	R3	R4	...
	R15		
PLFACT (open ended)	SID	B1	B2
	...	B_n	-1

2.3.2.7 DIT (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
TABDMP1	15	21	162
TABDMP2	25	22	163
TABDMP3	35	23	164
TABDMP4	45	24	165
TABLED1	1105	11	133
TABLED2	1205	12	134
TABLED3	1305	13	140
TABLED4	1405	14	141
TABLEM1	105	1	93
TABLEM2	205	2	94
TABLEM3	305	3	95
TABLEM4	405	4	96
TABLES1	3105	31	97
TABRND1	55	25	191
TABRND2	65	26	192
TABRND3	75	27	193
TABRND4	85	28	194

Card Type Formats:

TABDMP1 (open ended)	ID	0	0
	0	0	0
	0	0	f_1
	g_1	f_2	g_2
	...	f_n	g_n
	-1	-1	
TABDMP2 (open ended)	ID	X1	0
	0	0	0
	0	0	f_1
	g_1	f_2	g_2
	...	f_n	g_n
	-1	-1	
TABDMP3 (open ended)	ID	X1	X2
	0	0	0
	0	0	f_1
	g_1	f_2	g_2
	...	f_n	g_n
	-1	-1	
TABDMP4 (open ended)	ID	X1	X2
	X3	X4	0
	0	0	A_0
	A_1	A_2	...
	A_n	-1	

Card Type Formats Cont'd.:

TABLED1 (open ended)	ID	0	0
	0	0	0
	0	0	x_1
	y_1	x_2	y_2
	...	x_n	y_n
	-1	-1	
TABLED2 (open ended)	ID	X1	0
	0	0	0
	0	0	x_1
	y_1	x_2	y_2
	...	x_n	y_n
	-1	-1	
TABLED3 (open ended)	ID	X1	X2
	0	0	0
	0	0	x_1
	y_1	x_2	y_2
	...	x_n	y_n
	-1	-1	
TABLED4 (open ended)	ID	X1	X2
	X3	X4	0
	0	0	A_0
	A_1	A_2	...
	A_n	-1	
TABLEM1 (open ended)	ID	0	0
	0	0	0
	0	0	x_1
	y_1	x_2	y_2
	...	x_n	y_n
	-1	-1	
TABLEM2 (open ended)	ID	X1	0
	0	0	0
	0	0	x_1
	y_1	x_2	y_2
	...	x_n	y_n
	-1	-1	
TABLEM3 (open ended)	ID	X1	X2
	0	0	0
	0	0	x_1
	y_1	x_2	y_2
	...	x_n	y_n
	-1	-1	
TABLEM4 (open ended)	ID	X1	X2
	X3	X4	0
	0	0	A_0
	A_1	A_2	...
	A_n	-1	

Card Type Formats Cont'd.:

TABLES1 (open ended)	ID	0	0
	0	0	0
	0	0	x_1
	y_1	x_2	y_2
	...	x_n	y_n
	-1	-1	
TABRND1 (open ended)	ID	0	0
	0	0	0
	0	0	f_1
	g_1	f_2	g_2
	...	f_n	g_n
	-1	-1	
TABRND2 (open ended)	ID	X1	0
	0	0	0
	0	0	f_1
	g_1	f_2	g_2
	...	f_n	g_n
	-1	-1	
TABRND3 (open ended)	ID	X1	X2
	0	0	0
	0	0	f_1
	g_1	f_2	g_2
	...	f_n	g_n
	-1	-1	
TABRND4 (open ended)	ID	X1	X2
	X3	X4	0
	0	0	A_0
	A_1	A_2	...
	A_n	-1	

2.3.2.8 EDT (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
DEFØRM	104	1	81
DFØRM	204	2	161

Card Type Formats:

DEFØRM (3 words)	SID	EID	D
DFØRM (4 words)	SID D	EID	FE

2.3.2.9 DYNAMICS (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
DAREA	27	17	182
DELAY	37	18	183
DLØAD	57	5	123
DPHASE	77	19	184
EIGB	107	1	86
EIGC	207	2	87
EIGP	257	4	158
EIGR	307	3	85
EPØINT	707	7	124
FREQ	1307	13	126
FREQ1	1007	10	125
FREQ2	1107	11	166
NØLIN1	3107	31	127
NØLIN2	3207	32	128
NØLIN3	3307	33	129
NØLIN4	3407	34	130
RANDPS	2107	21	195
RANDT1	2207	22	196
RLØAD1	5107	51	131
RLØAD2	5207	52	132
SEQEP	5707	57	135
TF	6207	62	136
TIC	6607	66	137
TLØAD1	7107	71	138
TLØAD2	7207	72	139
TSTEP	8307	83	142

Card Type Formats:

DAREA (4 words)	SID	P	C
	A		
DELAY (4 words)	SID	P	C
	T		
DLØAD (open ended)	SID	S	S1
	L1	S2	L2
	...	S_n	L_n
	-1	-1	
DPHASE (4 words)	SID	P	C
	TH		
EIGB (18 words)	SID	METHØD (2 words)	L1
	L2	NEP	NDP
	NDN	E	NØRM (2 words)
	G	C	0
	0	0	0
	0		

Card Type Formats Cont'd.:

EIGC (open ended)	SID	METHØD (2 words)	NØRM (2 words)
	G	C	E
	α_{a1}	ω_{a1}	α_{b1}
	ω_{b1}	ℓ_1	N_{e1}
	N_{d1}	α_{a2}	ω_{a2}
	α_{b2}	ω_{b2}	ℓ_2
	N_{e2}	N_{d2}	...
	α_{an}	ω_{an}	α_{bn}
	ω_{bn}	ℓ_n	N_{en}
	N_{dn}	-1	-1
	-1	-1	-1
	-1	-1	
EIGP (4 words)	SID	α	ω
	M		
EIGR (18 words)	SID	METHØD (2 words)	F1
	F2	NE	ND
	NZ	E	NØRM (2 words)
	G	C	0
	0	0	0
	0		
EPØINT (1 word)	ID		
FREQ (open ended)	SID	F	F
	F	...	F
	-1		
FREQ1 (4 words)	SID	F1	DF
	NDF		
FREQ2 (4 words)	SID	F1	F2
	NF		
NØLIN1 (8 words)	SID	GI	CI
	S	GJ	CJ
	T	0	
NØLIN2 (8 words)	SID	GI	CI
	S	GJ	CJ
	GK	CK	
NØLIN3 (8 words)	SID	GI	CI
	S	GJ	CJ
	A	0	
NØLIN4 (8 words)	SID	GI	CI
	S	GJ	CJ
	A	0	

Card Type Formats Cont'd.:

RANDPS (6 words)	SID	J	K
	X	Y	TID
RANDT1 (4 words)	SID	N	TO
	TMAX		
RLØAD1 (6 words)	SID	L	M
	N	TC	TD
RLØAD2 (6 words)	SID	L	M
	N	TB	TP
SEQEP (2 words)	ID	SEQID	
TF (open ended)	SID	GD	CD
	BO	B1	B2
	G(1)	C(1)	AO(1)
	A1(1)	A2(1)	G(2)
	C(2)	AO(2)	A1(2)
	A2(2)	...	G(N)
	C(N)	AO(N)	A1(N)
	A2(N)	-1	-1
	-1	-1	-1
TIC (5 words)	SID	G	C
	UO	VO	
TLØAD1 (5 words)	SID	L	M
	O	TF	
TLØAD2 (10 words)	SID	L	M
	O	T1	T2
	F	P	C
	B		
TSTEP (open ended)	SID	N(1)	DT(1)
	NØ(1)	N(2)	DT(2)
	NØ(2)	...	N(N)
	DT(N)	NØ(N)	-1
	-1	-1	

2.3.2.10 MATPØØL (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
DMIG	114	1	120

Card Type Formats:

DMIG (open ended)	NAME (2 words)	0	IFØ	TIN	TØUT	Header Information for the matrix (9 words)
	0	0	0			
GJ	CJ	GI	CI	V*		
		GI	CI	V*		Non-zero terms of the first non-zero column
		.	.	.		
		.	.	.		
		.	.	.		
		GI	CI	V*		
		-1	-1			End of column indicators
GJ	CJ	GI	CI	V*		
		GI	CI	V*		Non-zero terms of the second non-zero column
		.	.	.		
		.	.	.		
		.	.	.		
		GI	CI	V*		
		-1	-1			End of column indicators
.						
.						
.						
GJ	CJ	GI	CI	V*		
		GI	CI	V*		Non-zero terms of the last non-zero column
		.	.	.		
		.	.	.		
		.	.	.		
		GI	CI	V*		
		-1	-1			End of column indicators
-1	-1					End of matrix indicators

*V may be 1, 2, or 4 words depending on TIN.

2.3.2.11 AXIC (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
AXIC	515	5	144
CCØNEAX	2315	23	146
FØRCEAX	2115	21	156
MØMAX	3815	38	157
MPCAX	4015	40	149
ØMITAX	4315	43	150
PØINTAX	4915	49	152
PRESAX	5215	52	154
RINGAX	5615	56	145
SECTAX	6015	60	153
SPCAX	6215	62	148
SUPAX	6415	64	151
TEMPAX	6815	68	155

Card Type Formats:

AXIC (2 words)	H	0	
CCØNEAX (4 words)	ID	PID	RA
	RB		
FØRCEAX (7 words)	SID	RID	HID
	S	FR	FP
	FZ		
MØMAX (7 words)	SID	RID	HID
	S	MR	MP
	MZ		
MPCAX (open ended)	SID	RID	HID
	C	A	RID
	HID	C	A
	...	RID	HID
	C	A	-1
	-1	-1	-1
ØMITAX (3 words)	RID	HID	C
PØINTAX (3 words)	ID	RID	PHI
PRESAX (6 words)	SID	P	RID1
	RID2	PHI1	PHI2
RINGAX (4 words)	ID	R	Z
	C		
SECTAX (5 words)	ID	RID	R
	PHI1	PHI2	
SPCAX (5 words)	SID	RID	HID
	C	V	
SUPAX (3 words)	RID	HID	C
TEMPAX (4 words)	SID	RID	PHI
	TEMP		

2.3.2.12 FØRCE (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
EDFIR	1513	15	160
ELDELE	1913	19	188
MASSC	3313	33	159
VECDN	8313	83	117
VECGP	8413	84	118
ZI	8913	89	113

Card Type Formats:

EDFIR (7 words)	SID	EID	FE
	EID	FE	EID
	FE		
ELDELE (7 words)	SID	EID	FE
	EID	FE	EID
	FE		
MASSC (8 words)	GP1	DF	GP2
	DF	MASS	GP3
	DF	MASS	
VECDN (5 words)	VID	GID	A
	B	C	
VECGP (5 words)	VID	G1	G2
	G3	G4	
ZI (5 words)	EID1	FE	EID2
	FE	Z	

2.3.3 Data Blocks Output From Module GP1

2.3.3.1 GPL (TABLE)

Description

Grid Point List.

First logical record contains a list of external grid and scalar numbers in internal sort. Second logical record contains pairs of external grid and scalar numbers and sequence numbers in internal sort.

Table Format

Record	Word	Item	
0		Header record	
1	1	External grid or scalar number	repeated for each grid or scalar point in model
2	1	External grid or scalar number	repeated for each grid or scalar point in model
	2	Sequence number	
3		End-of-file	

Notes

1. Internal is implied by word position in record one.
2. Sequence number = 1000 * external number unless replaced by a new sequence number on a SEQGP card.
3. All data words are integers.

Table Trailer

Word 1 = number of external grid points + number of scalar points.

Word 2-6 = zero.

2.3.3.2 EQEXIN (TABLE)

Description

Equivalence between external grid or scalar numbers and internal numbers.

First record contains pairs of external grid and scalar numbers and internal numbers in external sort. Second logical record contains pairs of external grid and scalar numbers and coded SIL numbers in external sort.

Table Format

Record	Word	Item	
0		Header record	
1	1	External grid or scalar number	repeated for each grid or scalar point in model
	2	Internal number	
2	1	External grid or scalar number	repeated for each grid or scalar point in model
	2	10*SIL number + code	
3		End-of-file	

Notes

1. All data words are integers.
2. Code = {1 for grid point; 2 for scalar point}

Table Trailer

Word 1 = number of grid points + number of scalar points.

Word 2-6 = zero.

2.3.3.3 GPDT (TABLE)

Description

Grid Point Definition Table.

One logical record contains list of all grid and scalar points with associated coordinate system and constraint information. List is in internal sort.

Table Format

Record	Word	Item	
0		Header record	
1	1	Internal number	repeated for each grid or scalar
	2	Coordinate system ID that defines x, y, z	
	3	$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ or $\begin{pmatrix} R \\ \theta \\ z \end{pmatrix}$ or $\begin{pmatrix} \rho \\ \theta \\ \phi \end{pmatrix}$ depending on defining coordinate system	
	4		
	5		
	6	Coordinate system ID for displacements	
	7	Constraint code	
2		End-of-file	

Notes

1. Words 3-5 are single precision floating point; all other words are integer.
2. Scalar points are identified by coordinate system ID = -1, and words 3-7 are all zero.
3. See description of the GRID bulk data card in the User's Manual for a definition of the constraint code.

Table Trailer

Word 1 = number of grid points + number of scalar points.

Word 2-6 = zero.

2.3.3.4 CSTM (TABLE)

Description

Coordinate System Transformation Matrices.

One logical record contains all coordinate system transformations. Transformation is from global to basic by the following formulation:

(1) rectangular

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_B = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_g + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

(2) cylindrical

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_B = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} R\cos\theta \\ R\sin\theta \\ z \end{pmatrix}_g + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

(3) spherical

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_B = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} \rho\sin\theta\cos\Phi \\ \rho\sin\theta\sin\Phi \\ \rho\cos\theta \end{pmatrix}_g + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

Table Format

Record	Word	Item
0		Header record
1	1	Coordinate system ID
	2	Coordinate system type { 1 = rectangular, 2 = cylindrical, 3 = spherical }
	3-5	t_1, t_2, t_3
	6-14	r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}
2		End-of-file

Record 1: repeated for each coordinate system

Notes

1. Coordinate system ID and coordinate system type are integers.
2. t_i and r_{ij} are single precision floating point.

Table Trailer

Word 1 = number of grid points + number of scalar points.

Word 2 = number of coordinate systems.

Word 3-6 = zero.

2.3.3.5 BGPDT (TABLE)

Description

Basic Grid Point Definition Table.

One logical record contains a list of all grid and scalar points in internal sort, with (for grid points) their x, y, z coordinates in the basic system along with a coordinate system ID for displacement computations.

Table Format

Record	Word	Item
0		Header record
1	1	Coordinate system ID } repeated for each
	2-4	x, y, z in basic system } grid or scalar point
2		End-of-file

Notes

1. Coordinate system ID is integer; x, y, z are single precision, floating point.
2. Scalar points are identified by coordinate system ID = -1, and x, y, z = 0.

Table Trailer

Word 1 = number of grid points + number of scalar points.

Word 2-6 = zero.

2.3.3.6 SIL (TABLE)

Description

Scalar Index List.

One logical record that contains a list of SIL numbers for each grid or scalar point. The list is in internal sort, therefore, internal number is implied by word position in the record. Definition of SIL numbers is as follows:

Let i = internal number, then

$$SIL_1 = 1,$$

$$SIL_{i+1} = \begin{cases} SIL_i + 6 & \text{if } i = \text{grid point} \\ SIL_i + 1 & \text{if } i = \text{scalar point} \end{cases}$$

Table Format

Record	Word	Item
0		Header record
1	1	SIL_1
	⋮	⋮
	n	SIL_n
2		End-of-file

Notes

SIL numbers are integers.

Table Trailer

Word 1 = number of grid points + number of scalar points.

Word 2 = degrees of freedom in the g-displacement set.

Word 3-6 = zero.

2.3.4 Data Blocks Output From Module GP2

2.3.4.1 ECT (TABLE)

Description

Element Connection Table.

The ECT contains one logical record for each element connection card type that has been input. Additionally, the ECT contains one logical record for GENEL elements if they have been input.

Table Format

The ECT is identical in format to data block GEØM2, output from module IFP. All external grid or scalar numbers are replaced by internal numbers. SPØINT data is not copied on the ECT.

Table Trailer

Identical to trailer on GEØM2 data block.

2.3.5 Data Blocks Output From Module PLTSET

2.3.5.1 PLTSETX (TABLE)

Description

User error messages related to the definition of element plot sets for the structure plotter.

Table Format

See the description of the MESSAGE table, section 2.3.5.5.

Note

PLTSETX is generated in subroutine SETINP.

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.5.2 PLTPAR (TABLE)

Description

Plot parameters and plot control table.

Table Format

Record	Word	Item
0		Header record
1 2 3 etc.		Duplicate of the plot control data block (PCDB) created in the IFP1 module except that all plot set definitions have been deleted.
Last		End-of-file

Note

PLTPAR is generated in subroutine SETINP.

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.5.3 GPSETS (TABLE)

Description

Grid point sets related to the element plot sets.

Table Format

Record	Word	Item
0		Header record
1	1-NSETS	Element plot set ID's (integer)
2-(NSETS+1)	1	Number of grid points in an element set
	2-(NGP+1)	Pointers to the grid points in this element set (integers) 1 If = 0, the grid point is not in this set 2 If ≠ 0, this is an internal index relative to only the grid points in this element set (if negative, this grid point is to be excluded when used to draw deformed shapes and vectors)
NSETS+2		End-of-file

Notes

1. NSETS = number of element sets
2. NGP = total number of structural grid points
3. GPSETS is generated in subroutines SETINP and CNSTRC

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.5.4 ELSETS (TABLE)

Description

Element plot set connection tables.

Table Format

Record	Group	Word	Item
0			Header record
1-NSETS	1-NTYPS	1	Number of grid points per element of a given element type (integer) Note: If less than 3 or negative, this element type does not define a closed area.
		2-(N+1)	Grid point indices relative to all grid points defining each element of this type (integers)
		N+2	Integer zero
NSETS+1			End-of-file

Notes

1. NSETS = number of element plot sets
2. NTYPS = number of element types represented in an element plot set
3. N = number of connection grid points for all elements of a given type in an element plot set
   = (number of grid points per element of a given type) X (number of elements of a given type in an element set)
4. ELSETS is generated in subroutine CNSTRC

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.5.5 MESSAGE (TABLE)

Description

Messages to be processed by the message writer module (PRTMSG). Each message may either be a physical or logical record. This data block is never really created as such, but is included so as to explain other data blocks such as PLØTX1, PLØTX2, etc., and is referenced in the Table Formats of these data blocks.

Table Format

Record	Word	Item
0		Header record

A given logical record in a given physical record can be of two alternate forms

A.

Record	Word	Item
i	j	If = -1, -2, -3, -4, -5, or -6, then the next 32 words is a new title for the 1st, 2nd, 3rd, 4th, 5th, or 6th lines on all printed pages to follow from this message table (integer)
	(j+1)-(j+32)	The 32 4-character BCD words for this title

B.

Record	Word	Item
i	j	NLIST = number of list items (integer)
	(j+1)-(j+NLIST)	List items (mixed mode)
	j+NLIST+1	NF = size of format to be used to print these list items (integer)
	(j+NLIST+2)-(j+NLIST+NF+1)	Format to be used to print this list (series of consecutive BCD characters)

2.3.6 Data Blocks Output From Module PLØT

2.3.6.1 PLØTX1 (TABLE)

Description

User messages from the plot module relative to the undeformed structural shapes

Table Format

See the description of the MESSAGE table, section 2.3.5.5

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.6.2 PLØTX2 (TABLE)

Description

User messages from the plot module relative to the deformed structural shapes generated in the statics analysis

Table Format

See the description of the MESSAGE table, section 2.3.5.5

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.7 Data Blocks Output From Module GP3

2.3.7.1 SLT (TABLE)

Description

Static Loads Table.

The header record of the SLT contains a sorted list of all unique load set ID's contained on static load cards except the LØAD card itself. The n logical records that follow the header record comprise all static loads data belonging to each of the n load sets, one logical record per load set. The (n+1)st logical record contains all LØAD cards (if any).

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3	I	Load set ID_1
	⋮		⋮
	2+n	I	Load set ID_n
1	1,2	I	Load card type, m
	3		Load data as function of load card type... repeated m times
	⋮		(repeated for each different load card type belonging to load set number 1)
	2+g		
⋮			
n			Same format as record 1 Data belongs to load set number n
n+1	1,2	I,R	Load set ID, scale factor
	3-4	I,R	Load set ID, scale factor (repeated for each LØAD card)
	⋮		
	2k+3,2k+4	I	-1, -1
n+2			End-of-file

Notes

1. The SLT is generated in subroutine GP3A.
2. Card type ID's and format for data for each bulk data card type are as follows:

   3 = FØRCE1

Word	Type	Item
1	I	Internal grid number
2	R	Signed magnitude of applied load
3-4	I	Internal grid numbers of grid points that define direction

5 = FØRCE2

Word	Type	Item
1	I	Internal grid number
2	R	Signed magnitude of applied load
3-6	I	Internal grid numbers of grid points that define direction

1 = FØRCE

Word	Type	Item
1	I	Internal grid number
2	I	Coordinate system ID
3	R	Signed scale factor for applied force
4-6	R	Force components

8 = GRAV

Word	Type	Item
1	I	Coordinate system ID
2	R	Gravity vector scale factor
3-5	R	Gravity vector components

4 = MØMENT1

See FØRCE1, substituting "moment" for "force".

6 = MØMENT2

See FØRCE2, substituting "moment" for "force".

2 = MØMENT

See FØRCE, substituting "moment" for "force".

9 = PLØAD

Word	Type	Item
1	R	Pressure
2-5	I	Internal grid numbers

7 = SLØAD

Word	Type	Item
1	I	Internal scalar number
2	R	Applied load

10 = RFØRCE

Word	Type	Item
1	I	Internal grid number
2	I	Coordinate system ID
3	R	Scale factor
4-6	R	Components of rotation direction vector

11 = PRESAX

Word	Type	Item
1	R	Pressure value
2-3	I	Ring ID's
4-5	R	Azimuthal angles
6	I	Number of harmonics

3. With the exception of GRAV and PLØAD card types, data of a given card type within a logical record is in sort on internal grid (or scalar) number at which the load is applied.

4. If no LØAD cards have been input, the (n+1)st record does not exist.

Table Trailer

Word 1 = number of load sets.

Word 2-6 = zero.

2.3.7.2 GPTT (TABLE)

Description

Grid Point Temperature Table.

The header record of the GPTT contains sorted triples of temperature set ID, default temperature, and flag. For each temperature set for which temperature data is defined at the grid points, a logical record in the GPTT is present.

Table Format

Record	Word	Item	
0	1-2	Data block name (BCD)	
	3	Temperature set ID, (integer)	repeated for each temperature set
	4	Default temperature (floating point) -1 if no default temperature defined (integer)	
	5	0 if only default temperature for set (integer) >0 record number of temperature data for set (integer)	
1	1	Internal grid number (integer)	repeated for every grid point at which a temperature is defined
	2	Temperature (floating point)	
⋮			
k		Same format as record 1	
k+1		End-of-file	

Notes

1. The GPTT is generated in subroutine GP3B.
2. A temperature set may be defined as consisting only of a default temperature that applies to all grid points.
3. A default temperature (if defined) is to apply to all grid points for which a temperature has not been defined.
4. Each data record is in sort on internal grid number.

Table Trailer

Word 1 = 7.

Word 2-6 = undefined.

2.3.8 Data Blocks Output From Module TA1

2.3.8.1 EST (TABLE)

Description

Element Summary Table.

The EST is a collection of data for all elements of the structural model. It contains one logical record for each element type. For each element: connection data, properties data, basic grid point data and the element temperature are grouped. General elements and elements that belong to super elements are not included in the EST.

Table Format

Record	Word	Type	Item		
0			Header record		
1	1	I	Element type		repeated for each element type
	2-i+1		ECT section	repeated for each element	
	i+2-i+j+1		EPT section		
	i+j+2-i+j+k+1		BGPDT section		
	i+j+k+2		Element temperature		
n+1			End-of-file		

Notes

1. i = number of words in ECT section.
   j = number of words in EPT section.
   k = number of words in BGPDT section.
2. The number of records in the EST corresponds to the number of separate element types in the model.
3. The EST is generated in subroutine TA1A.

Summary of EST Formats

Element Type	Mnemonic	ECT Section Number of Words	EPT Section number of Words	BGPDT Section Number of Words	Element Temperature	Total Words Per Element
1	ROD	3	5	8	Yes	17
2	BEAM	19	19	8	Yes	47
3	TUBE	3	4	8	Yes	16
4	SHEAR	5	3	16	Yes	25
5	TWIST	5	3	16	Yes	25
6	TRIA1	5	9	12	Yes	27
7	TRBSC	5	7	12	Yes	25
8	TRPLT	5	7	12	Yes	25
9	TRMEM	5	3	12	Yes	21
10	CONROD	8	0	8	Yes	17
11	ELAS1	5	3	0	No	8
12	ELAS2	8	0	0	No	8
13	ELAS3	3	3	0	No	6
14	ELAS4	4	0	0	No	4
15	QDPLT	6	7	16	Yes	30
16	QDMEM	6	3	16	Yes	26
17	TRIA2	5	3	12	Yes	21

Element Type	Mnemonic	ECT Section Number of Words	EPT Section Number of Words	BGPDT Section Number of Words	Element Temperature	Total Words Per Element
18	QUAD2	6	3	16	Yes	26
19	QUAD1	6	9	16	Yes	32
20	DAMP1	5	1	0	No	6
21	DAMP2	6	0	0	No	6
22	DAMP3	3	1	0	No	4
23	DAMP4	4	0	0	No	4
24	VISC	3	2	8	Yes	14
25	MASS1	5	1	0	No	6
26	MASS2	6	0	0	No	6
27	MASS3	3	1	0	No	4
28	MASS4	4	0	0	No	4
29	CØNM1	24	0	4	Yes	29
30	CØNM2	13	0	4	Yes	18
31	PLØTEL	3	0	8	Yes	12
32	REACT	19	0	4	Yes	24
33	QUAD3	7	1	16	Yes	24
34	BAR	15	18	8	Yes	42
35	CONEAX	3	23	8	Yes	35
36	TRIARG	6	0	12	Yes	19
37	TRAPRG	7	0	16	Yes	24
38	TØRDRG	6	3	2	Yes	18

Detailed EST Formats

ECT section for element type = 2:

Word	Type	Item
1	I	Element ID
2-3	I	SIL numbers for grid points 1, 2
4-6	R	x, y, z (orientation vector)
7	I	Coordinate system ID for x, y, z
8-9	I	P_a, P_b
10-12	R	Z_a^1, Z_a^2, Z_a^3
13-15	R	Z_b^1, Z_b^2, Z_b^3
16-19	R	g_1, g_2, g_3, g_4

ECT Section for element type = 1, 3, 24, 31:

Word	Type	Item
1	I	Element ID
2-3	I	SIL numbers for grid points 1, 2

ECT section for element type = 4, 5:

Word	Type	Item
1	I	Element ID
2-5	I	SIL numbers for grid points 1, 2, 3, 4

ECT section for element type = 6, 7, 8, 9, 17:

Word	Type	Item
1	I	Element ID
2-4	I	SIL numbers for grid points 1, 2, 3
5	R	θ (degrees)

ECT section for element type = 15, 16, 18, 19:

Word	Type	Item
1	I	Element ID
2-5	I	SIL numbers for grid points 1, 2, 3, 4
6	R	θ (degrees)

ECT section for element type = 10:

Word	Type	Item
1	I	Element ID
2-3	I	SIL numbers for grid points 1, 2
4	I	Material ID
5	R	A
6	R	J
7	R	C
8	R	non-structural mass (nsm)

ECT section for element type = 11, 20, 25:

Word	Type	Item
1	I	Element ID
2-3	I	SIL numbers for grid points 1, 2
4-5	I	Component codes for grid points 1, 2

ECT section for element type = 12:

Word	Type	Item
1	I	Element ID
2	R	Value
3-4	I	SIL numbers for grid points 1, 2
5-6	I	Component codes for grid points 1, 2
7-8	R	g_e, S

ECT section for element type = 13, 22, 27:

Word	Type	Item
1	I	Element ID
2-3	I	SIL numbers for scalar points 1, 2

ECT section for element type = 14, 23, 28:

Word	Type	Item
1	I	Element ID
2	R	Value
3-4	I	SIL numbers for scalar points 1, 2

ECT section for element type = 21, 26:

Word	Type	Item
1	I	Element ID
2	R	Value
3-4	I	SIL numbers for grid points 1, 2
5-6	I	Component codes for grid points 1, 2

ECT section for element type = 29:

Word	Type	Item
1	I	Element ID
2	I	SIL number for grid point
3	I	Coordinate system ID
4-24	R	m_{11}, m_{21}, m_{22}, m_{31}, etc., (6x6 symmetric matrix)

ECT section for element type = 30:

Word	Type	Item
1	I	Element ID
2	I	SIL number for grid point
3	I	Coordinate system ID
4	R	m
5-7	R	x_1, x_2, x_3
8-13	R	I_{11}, I_{21}, I_{22}, I_{31}, I_{32}, I_{33}

ECT section for element type = 34:

Word	Type	Item
1	I	Element ID
2-3	I	SIL values for grid points 1, 2
4-6	R	X_1, X_2, X_3
7	I	Coordinate system ID for X_1, X_2, X_3
8-9	I	P_a, P_b
10-12	R	Z_a^1, Z_a^2, Z_a^3
13-15	R	Z_b^1, Z_b^2, Z_b^3

ECT section for element type = 35:

Word	Type	Item
1	I	Element ID
2-3	I	SIL values for rings 1, 2

ECT section for element type = 36:

Word	Type	Item
1	I	Element ID
2-4	I	SIL values for grid points 1, 2, 3
5	R	θ (degrees)
6	I	Material ID

ECT section for element type = 37:

Word	Type	Item
1	I	Element ID
2-5	I	SIL values for grid points 1, 2, 3, 4
6	R	θ (degrees)
7	I	Material ID

ECT section for element type = 38:

Word	Type	Item
1	I	Element ID
2-3	I	SIL values for grid points 1, 2
4-5	R	A_1, A_2
6		Not defined

EPT section for element type = 1:

Word	Type	Item
1	I	Material ID
2	R	A
3	R	J
4	R	C
5	R	nsm

EPT section for element type = 2:

Word	Type	Item
1	I	Material ID
2	R	A
3-4	R	I_1, I_2
5	R	J
6	R	nsm
7	I	Force Element Code (FE)
8-9	R	C_1, C_2
10-11	R	D_1, D_2
12-13	R	E_1, E_2
14-15	R	F_1, F_2
16-17	R	K_1, K_2
18	R	I_{12}
19		not defined

EPT section for element type = 3:

Word	Type	Item
1	I	Material ID
2	R	O.D.
3	R	t
4	R	nsm

EPT section for element type = 4, 5, 9, 16, 17, 18:

Word	Type	Item
1	I	Material ID
2	R	t
3	R	nsm

EPT section for element type = 6, 19:

Word	Type	Item
1	I	Material ID for membrane
2	R	t_1
3	I	Material ID for bending
4	R	I
5	I	Material ID for transverse shear
6	R	t_2
7	R	nsm
8-9	R	Z_1, Z_2

EPT section for element type = 7, 8, 15:

Word	Type	Item
1	I	Material ID for bending
2	R	I
3	I	Material ID for transverse shear
4	R	t_2
5	R	nsm
6-7	R	Z_1, Z_2

EPT section for element type = 11, 13:

Word	Type	Item
1	R	K
2	R	g_e
3	R	s

EPT section for element type = 20, 22:

Word	Type	Item
1	R	B_e

EPT section for element type = 24:

Word	Type	Item
1	R	C_1
2	R	C_2

EPT section for element type = 25, 27:

Word	Type	Item
1	R	M_e

EPT section for element type = 33:

Word	Type	Item
1	I	Super element property ID

EPT section for element type = 34:

Word	Type	Item
1	I	Material ID
2	R	A
3-4	R	I_1, I_2
5	R	J
6	R	nsm
7	I	FE (Force Method only)
8-9	R	C_1, C_2
10-11	R	D_1, D_2
12-13	R	E_1, E_2
14-15	R	F_1, F_2
16-17	R	K_1, K_2
18	R	I_{12}

EPT section for element type = 35:

Word	Type	Item
1	I	Material ID for membrane
2R	R	T_1
3	I	Material ID for bending
4	R	I
5	I	Material ID for transverse shear
6	R	T_2
7	R	nsm
8-9	R	Z_1, Z_2
10-23	R	Φ_i, i = 1, 14

EPT section for element type = 38:

Word	Type	Item
1	I	Material ID
2	R	TM
3	R	TF

Table Trailer

Word 1 = number of elements in model.

Word 2-6 = are undefined.

2.3.8.2 GEI (TABLE)

Description

General Element Input.

The GEI contains one logical record for each general element in the model.

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	ID for general element
	2	I	n = number of elements in U_I list
	3	I	m = number of elements in U_D list
	4	I	SIL value for first U_I
	⋮		
	3+n	I	SIL value for n^{th} U_I
	4+n	I	SIL value for first U_D (not present if m = 0)
	⋮		⋮
	3+n+m	I	SIL value for m^{th} U_D
	4+n+m ⋮ $3+n+m+n^2$	R	Elements of Z matrix
	$4+n+m+n^2$ ⋮ $3+n+m+n^2+nm$	R	Elements of S matrix
2			Same format as record 1
⋮			
k			Same format as record 1
k+1			End-of-file

Table Trailer

Word 1 = number of general elements in the model.

Words 2-6 = zero.

2.3.8.3 ECPT (TABLE)

Description

Element Connection and Properties Table.

The ECPT is essentially the EST in a different sort. The ECPT contains one logical record for each grid or scalar point of the model. Each logical record contains Element Summary Table data for each element connected to the grid or scalar point.

Table Format

Record	Word	Item	
0		Header record	
1	1	SIL number for "pivot" grid or scalar point (integer)	repeated for each grid or scalar in the model
	2	Element type	repeated for each element connected to the pivot
	3-i+2	ECT section	
	i+3-i+j+2	EPT section	
	i+j+3-i+j+k+2	BGPDT section	
	i+j+k+3	Element temperature	
n+1		End-of-file	

Notes

1. Detailed formats are given in the EST writeup (see section 2.3.8.1).
2. If no elements are connected to a grid or scalar point, the record contains only one word.

Table Trailer

Word 1 = 7

Word 2-6 = are undefined

2.3.8.4 GPCT (TABLE)

Description

Grid Point Connection Table.

The GPCT is a condensation of the ECPT. It contains one logical record for each grid or scalar point of the model. Each logical record contains a list of all grid or scalar points that are connected (by means of structural elements) to the pivot grid or scalar point.

Table Format

Record	Word	Item	
0		Header record	
1	1	± SIL number for pivot grid or scalar point (integer)	repeated for each grid or scalar in the model
	2	m = number of connected points (integer)	
	3-2+m	Sorted list of SIL numbers of connected points	
n+1		End-of-file	

Notes

1. If the SIL number for the pivot (first word) < 0, then the pivot is a scalar point.
2. If no elements are connected to the pivot (and therefore no other grid or scalar points), the record contains only one word.

Table Trailer

Word 1 = 7

Word 2-6 = are undefined

2.3.9 Data Blocks Output From Module SMA1

2.3.9.1 KGGX (MATRIX)

Description

$[K^{x}_{gg}]$ - Partition of stiffness matrix exclusive of general elements - g set.

Matrix Trailer

Number of columns = g
Number of rows = g
Form = symmetric
Type = real double precision

2.3.9.2 K4GG (MATRIX)

Description

$[K^{4}_{gg}]$ - Partition of structural damping matrix - g set.

Matrix Trailer

Number of columns = g
Number of rows = g
Form = symmetric
Type = real double precision

2.3.9.3 GPST (TABLE)

Description

Grid Point Singularity Table

Table Format

Record	Word	Item	
0		Header record	
1	1	Order of singularity (1, 2, or 3)	Repeated for each singularity
	2	N = number of SIL numbers that follow	
	3	SIL_1	
	4	SIL_2	
	.		
	.		
	.		
	2+N	SIL_N	
2		End-of-file	

Note

All entries are integers.

Table Trailer

Word 1 = undefined
Word 2 = 0
Word 3 = 1
Word 4 = 2
Word 5 = 1
Word 6 = 0

2.3.10 Data Blocks Output From Module SMA2

2.3.10.1 MGG (MATRIX)

Description

$[M_{gg}]$ - Partition of mass matrix - g set.

Matrix Trailer

Number of columns	=	g
Number of rows	=	g
Form	=	symmetric
Type	=	real double precision

2.3.10.2 BGG (MATRIX)

Description

$[B_{gg}]$ - Partition of damping matrix - g set.

Matrix Trailer

Number of columns	=	g
Number of rows	=	g
Form	=	symmetric
Type	=	real double precision

2.3.11 Data Blocks Output From Module GPWG.

2.3.11.1 ØGPWG (TABLE)

Description

Grid Point Weight Generator Øutput Table.

Table Format

Record	Word	Type	Item
0			Header record
1			ØFP ID record
	1	I	1
	2	I	13
	3	I	External ID of grid point about which moments and interias were calculated. If External ID = 0 the basic origin was used.
	4-9		Not defined.
	10	I	98
	11-50		Not defined.
	51-146	B	96 words of title, subtitle, and label from /ØUTPUT/
2			ØFP data record
	1-36	R	[MO] 6x6 moment matrix
	37-45	R	[S] 3x3 matrix
	46-49	R	Mx, Xx, Yx, Zx
	50-53	R	My, Xy, Yy, Zy
	54-57	R	Mz, Xz, Yz, Zz
	58-66	R	Inertia matrix (3x3)
	67-69	R	Principal inertias
	70-78	R	Q matrix (3x3)
3			End-of-file

Table Trailer

Word 1 = 0

Word 2 = nonzero.

Words 3-6 = 0

2.3.12 Data Blocks Output From Module SMA3

2.3.12.1 KGG (MATRIX)

Description

$[K_{gg}]$ - Partition of stiffness matrix - g set. Contains contributions from all elements in the model, including general elements.

Matrix Trailer

Number of columns	=	g
Number of rows	=	g
Form	=	symmetric
Type	=	real double precision

2.3.12.2 KGGL (MATRIX)

Description

$[K^{\ell}_{gg}]$ - Partition of the stiffness matrix of linear elements - g set. Contains contributions from all linear elements of the model including general elements. Used only in Piecewise Linear Analysis.

Matrix Trailer

Number of columns	=	g
Number of rows	=	g
Form	=	symmetric
Type	=	real double precision

2.3.13 Data Blocks Output From Module GP4

2.3.13.1 RG (MATRIX)

Description

$[R_g]$ - Multipoint constraint equations matrix.

Matrix Header

Number of columns = g
Number of rows = m
Form = rectangular
Type = real single precision

2.3.13.2 YS (MATRIX)

Description

$\{Y_s\}$ - Constrained displacement vector - s set.

Matrix Trailer

Number of columns = 1
Number of rows = s
Form = rectangular
Type = real single precision

2.3.13.3 USET (TABLE)

Description

Displacement set definitions table.

USET contains one logical record. Each word corresponds to each degree of freedom in the g-displacement set (in internal order) and contains ones in specified bit positions indicating the displacement sets to which the degree of freedom belongs.

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3	I	SPC set ID
	4	I	MPC set ID
1	1	L	Mask for first degree of freedom
	⋮		⋮
	n	L	Mask for n^{th} degree of freedom
2			End-of-file

Notes

1. Bit positions for the various displacement sets are defined as follows:

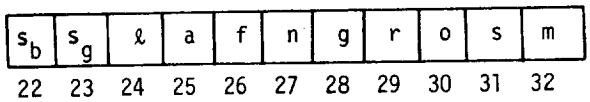


Table Trailer

Word 1 = zero.

Word 2 = degrees of freedom in the g-displacement set (LUSET).

Word 3 = zero.

Word 4 = logical "or" of all USET masks.

Word 5 = zero.

Word 6 = zero.

2.3.14 Data Blocks Output From Module GPSP.

2.3.14.1 ØGPST (TABLE)

Description

Unremoved Grid Point Singularities.

Table Format

Record	Word	Type	Item
0			Header record
1			ØFP ID record
	1	I	0
	2	I	8
	3-9		Not defined
	10	I	12
	11-50		Not defined
	51-146	B	96 words of title, subtitle, and label from /ØUTPUT/
2			ØFP data record
	1	I	External grid point ID
	2	I	Scalar point flag
	3	I	Singularity order
	4-6	I	Strongest singularity components
	7-9	I	Next strongest singularity components
	10-12	I	Weakest singularity components

*Note: The above 12 words are repeated in record 2 for each grid point with an unremoved singularity.

3			End-of-file

Table Trailer

Word 1 = 8

Word 2-6 = 0

2.3.15 Data Blocks Output From Module MCE1

2.3.15.1 GM (MATRIX)

Description

$[G_m]$ - Multipoint constraint transformation matrix - m set.

Matrix Header

Number of columns	=	n
Number of rows	=	m
Form	=	rectangular
Type	=	real double precision

2.3.16 Data Blocks Output From Module MCE2

2.3.16.1 KNN (MATRIX)

Description

$[K_{nn}]$ - Partition of stiffness matrix - n set.

Matrix Trailer

Number of columns	=	n
Number of rows	=	n
Form	=	symmetric
Type	=	real double precision

2.3.16.2 MNN (MATRIX)

Description

$[M_{nn}]$ - Partition of mass matrix - n set.

Matrix Trailer

Number of columns	=	n
Number of rows	=	n
Form	=	symmetric
Type	=	real double precision

2.3.16.3 KDNN (MATRIX)

Description

$[K^{d}_{nn}]$ - Partition of differential stiffness matrix - n set.

Matrix Trailer

Number of columns	=	n
Number of rows	=	n
Form	=	symmetric
Type	=	real double precision

2.3.16.4 BNN (MATRIX)

Description

$[B_{nn}]$ - Partition of damping matrix - n set.

Matrix Trailer

Number of columns	=	n
Number of rows	=	n
Form	=	symmetric
Type	=	real double precision

2.3.16.5 K4NN (MATRIX)

Description

$[K^4_{nn}]$ - Partition of the structural damping matrix - n set.

Matrix Trailer

Number of columns	=	n
Number of rows	=	n
Form	=	symmetric
Type	=	real double precision

2.3.17 Data Blocks Output From Module SCE1

2.3.17.1 KFF (MATRIX)

Description

$[K_{ff}]$ - Partition of stiffness matrix after single-point constraints have been removed - f set.

Matrix Trailer

Number of columns = f
Number of rows = f
Form = symmetric
Type = real double precision

2.3.17.2 KFS (MATRIX)

Description

$[K_{fs}]$ - Partition of stiffness matrix after single-point constraints have been removed.

Matrix Trailer

Number of columns = s
Number of rows = f
Form = rectangular
Type = real double precision

2.3.17.3 KSS (MATRIX)

Description

$[K_{ss}]$ - Partition of stiffness matrix after single-point constraints have been removed - s set.

Matrix Trailer

Number of columns = s
Number of rows = s
Form = symmetric
Type = real double precision

2.3.17.4 MFF (MATRIX)

Description

$[M_{ff}]$ - Partition of mass matrix after single-point constraints have been removed - f set.

Matrix Trailer

Number of columns = f
Number of rows = f
Form = symmetric
Type = real double precision

2.3.17.5 KDFF (MATRIX)

Description

$[K^d_{ff}]$ - Partition of differential stiffness matrix - f set.

Matrix Trailer

Number of columns = f
Number of rows = f
Form = symmetric
Type = real double precision

2.3.17.6 KDFS (MATRIX)

Description

$[K^d_{fs}]$ - Partition of differential stiffness matrix.

Matrix Trailer

Number of columns = s
Number of rows = f
Form = rectangular
Type = real double precision

2.3.17.7 KDSS (MATRIX)

Description

$[K^d_{ss}]$ - Partition of differential stiffness matrix - s set.

Matrix Trailer

Number of columns = s
Number of rows = s
Form = symmetric
Type = real double precision

2.3.17.8 BFF (MATRIX)

Description

$[B_{ff}]$ - Partition of damping matrix after single point constraints have been removed - f set.

Matrix Trailer

Number of columns = f
Number of rows = f
Form = symmetric
Type = real double precision

2.3.17.9 K4FF (MATRIX)

Description

$[K^4_{ff}]$ - Partition of structural damping matrix with single-point constraints removed - f set.

Matrix Trailer

Number of columns	=	f
Number of rows	=	f
Form	=	symmetric
Type	=	real double precision

2.3.18 Data Blocks Output From Module SMP1

2.3.18.1 GØ (MATRIX)

Description

$[G_o]$ - Structural matrix partitioning transformation matrix.

Matrix Trailer

Number of columns = a
Number of rows = o
Form = rectangular
Type = real double precision

2.3.18.2 KAA (MATRIX)

Description

$[K_{aa}]$ - Partition of stiffness matrix - a set.

Matrix Trailer

Number of columns = a
Number of rows = a
Form = symmetric
Type = real double precision

2.3.18.3 KØØB (MATRIX)

Description

$[\overline{K}_{oo}]$ - Partition of stiffness matrix - o set.

Matrix Trailer

Number of columns = o
Number of rows = o
Form = symmetric
Type = real double precision

2.3.18.4 LØØ (MATRIX)

Description

$[L_{oo}]$ - Lower triangular factor of KØØB - o set.

Matrix Trailer

Number of columns = o
Number of rows = o
Form = lower triangular
Type = real double precision

2.3.18.5 UØØ (MATRIX)

Description

$[U_{oo}]$ - Upper triangular factor of KØØB - o set.

Matrix Trailer

Number of columns	=	o
Number of rows	=	o
Form	=	upper triangular
Type	=	real double precision

Note

This matrix is not a standard upper triangular factor. Its format is acceptable only to subroutine FBS.

2.3.18.6 MAA (MATRIX)

Description

$[M_{aa}]$ - Partition of mass matrix - a set.

Matrix Trailer

Number of columns	=	a
Number of rows	=	a
Form	=	symmetric
Type	=	real double precision

2.3.18.7 MØØB (MATRIX)

Description

$[\overline{M}_{oo}]$ - Partition of mass matrix - o set.

Matrix Trailer

Number of columns	=	o
Number of rows	=	o
Form	=	symmetric
Type	=	real double precision

2.3.18.8 MØAB (MATRIX)

Description

$[\overline{M}_{oa}]$ - Partition of mass matrix.

Matrix Trailer

Number of columns = a
Number of rows = o
Form = rectangular
Type = real double precision

2.3.18.9 BAA (MATRIX)

Description

$[B_{aa}]$ - Partition of damping matrix - a set.

Matrix Trailer

Number of columns = a
Number of rows = a
Form = symmetric
Type = real double precision

2.3.18.10 K4AA (MATRIX)

Description

$[K^4_{aa}]$ - Partition of structural damping matrix - a set.

Matrix Trailer

Number of columns = a
Number of rows = a
Form = symmetric
Type = real double precision

2.3.19 Data Blocks Output From Module RBMG1

2.3.19.1 KLL (MATRIX)

Description

$[K_{\ell\ell}]$ - Partition of stiffness matrix - ℓ set.

Matrix Trailer

Number of columns	=	ℓ
Number of rows	=	ℓ
Form	=	symmetric
Type	=	real double precision

2.3.19.2 KLR (MATRIX)

Description

$[K_{\ell r}]$ - Partition of stiffness matrix

Matrix Trailer

Number of columns	=	ℓ
Number of rows	=	r
Form	=	rectangular
Type	=	real double precision

2.3.19.3 KRR (MATRIX)

Description

$[K_{rr}]$ - Partition of stiffness matrix - r set.

Matrix Trailer

Number of columns	=	r
Number of rows	=	r
Form	=	symmetric
Type	=	real double precision

2.3.19.4 MLL (MATRIX)

Description

$[M_{\ell\ell}]$ - Partition of mass matrix - ℓ set.

Matrix Trailer

Number of columns	=	ℓ
Number of rows	=	ℓ
Form	=	symmetric
Type	=	real double precision

2.3.19.5 MLR (MATRIX)

Description

$[M_{\ell r}]$ - Partition of mass matrix.

Matrix Trailer

Number of columns	=	ℓ
Number of rows	=	r
Form	=	rectangular
Type	=	real double precision

2.3.19.6 MRR (MATRIX)

Description

$[M_{rr}]$ - Partition of mass matrix - r set.

Matrix Trailer

Number of columns	=	r
Number of rows	=	r
Form	=	symmetric
Type	=	real double precision

2.3.20 Data Blocks Output From Module RBMG2

2.3.20.1 LLL (MATRIX)

Description

$[L_{\ell\ell}]$ - Lower triangular factor of KLL - ℓ set.

Matrix Trailer

Number of columns	=	ℓ
Number of rows	=	ℓ
Form	=	lower triangular
Type	=	real double precision

2.3.20.2 ULL (MATRIX)

Description

$[U_{\ell\ell}]$ - Upper triangular factor of KLL - ℓ set.

Matrix Trailer

Number of columns	=	ℓ
Number of rows	=	ℓ
Form	=	upper triangular
Type	=	real double precision

Note

This matrix is not a standard upper triangular factor. Its format is acceptable only to subroutine FBS.

2.3.20.3 LBLL (MATRIX)

Description

$[L^{b}_{\ell\ell}]$ - Lower triangular factor of KBLL - ℓ set.

Matrix Trailer

Number of columns	=	ℓ
Number of rows	=	ℓ
Form	=	lower triangular
Type	=	real double precision

2.3.20.4 UBLL (MATRIX)

Description

$[U^{b}_{\ell\ell}]$ - Upper triangular factor of KBLL - ℓ set.

Matrix Trailer

Number of columns	=	ℓ
Number of rows	=	ℓ
Form	=	upper triangular
Type	=	real double precision

Note

This matrix is not a standard upper triangular factor. Its format is acceptable only to subroutine FBS.

2.3.21 Data Blocks Output From Module RBMG3

2.3.21.1 DM (MATRIX)

Description

[D] - Rigid body transformation matrix.

Matrix Trailer

Number of columns	=	ℓ
Number of rows	=	r
Form	=	rectangular
Type	=	real double precision

2.3.22 Data Blocks Output From Module RBMG4

2.3.22.1 MR (MATRIX)

Description

$[m_r]$ - Rigid body mass matrix - r set.

Matrix Trailer

Number of columns	=	r
Number of rows	=	r
Form	=	symmetric
Type	=	real double precision

2.3.23 Data Blocks Output From Module SSG1.

2.3.23.1 PG (MATRIX)

Description

$[P_g]$ - Static load vector matrix giving static loads - g set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	g
Form	=	rectangular
Type	=	real single precision

2.3.23.2 PG1 (MATRIX)

Description

$[P_g^1]$ - Static load vector giving static loads for Piecewise Linear Analysis problem - g set.

Matrix Trailer

Number of columns	=	1
Number of rows	=	g
Form	=	rectangular
Type	=	real single precision

2.3.24 Data Blocks Output From Module SSG2

2.3.24.1 QR (MATRIX)

Description

$[q_r]$ - Determinate support forces matrix - r set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	r
Form	=	rectangular
Type	=	real single precision

2.3.24.2 PØ (MATRIX)

Description

$[P_o]$ - Partition of the load vector matrix giving loads due to static force - o set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	o
Form	=	rectangular
Type	=	real single precision

2.3.24.3 PS (MATRIX)

Description

$[P_s]$ - Partition of load vector matrix giving loads in s set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	s
Form	=	rectangular
Type	=	real single precision

2.3.24.4 PL (MATRIX)

Description

$[P_\ell]$ - Partition of the load vector matrix giving static loads on ℓ set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	ℓ
Form	=	rectangular
Type	=	real single precision

2.3.25 Data Blocks Output From Module SSG3

2.3.25.1 ULV (MATRIX)

Description

$[u_{\ell}]$ - Partition of the displacement vector matrix giving displacements - ℓ set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	ℓ
Form	=	rectangular
Type	=	real double precision

2.3.25.2 UØØV (MATRIX)

Description

$[u_o^o]$ - Partition of the displacement vector matrix giving displacements in the o set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	o
Form	=	rectangular
Type	=	real double precision

2.3.25.3 RULV (MATRIX)

Description

$[\delta P_{\ell}]$ - Residual vector matrix for the ℓ set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	ℓ
Form	=	rectangular
Type	=	real single precision

2.3.25.4 RUØV (MATRIX)

Description

$\{\delta P_o\}$ - Residual vector matrix for the o set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	o
Form	=	rectangular
Type	=	real single precision

2.3.25.5 UBLV (MATRIX)

Description

$[u_{\ell}^{b}]$ - Partition of the differential stiffness displacement vector - ℓ set.

Matrix Trailer

Number of columns = 1
Number of rows = ℓ
Form = rectangular
Type = real double precision

2.3.25.6 RUBLV (MATRIX)

Description

$[\delta P_{\ell}^{b}]$ - Differential stiffness residual vector - ℓ set.

Matrix Trailer

Number of columns = 1
Number of rows = ℓ
Form = rectangular
Type = real single precision

2.3.26 Data Blocks Output From Module SSG4.

2.3.26.1 PLI (MATRIX)

Description

$[P_{\ell}^{i}]$ - Partition of load vector for inertia relief matrix giving loads due to static + inertial forces on ℓ set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	ℓ
Form	=	rectangular
Type	=	real single precision

2.3.26.2 PØI (MATRIX)

Description

$[P_{o}^{i}]$ - Partition of load vector for inertia relief matrix giving loads due to inertial force + static forces on o set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	o
Form	=	rectangular
Type	=	real single precision

2.3.27 Data Blocks Output From Module SDR1

2.3.27.1 UGV (MATRIX)

Description

$[u_g]$ - Displacement vector matrix giving displacements in the g set.

Matrix Trailer

Number of columns	=	number of subcases in CASECC
Number of rows	=	g
Form	=	rectangular
Type	=	real single precision

2.3.27.2 PGG (MATRIX)

Description

$[P_g]$ - Static load vector appended to include all boundary conditions - g set.

Matrix Trailer

Number of columns	=	number of subcases in CASECC
Number of rows	=	g
Form	=	rectangular
Type	=	real single precision

2.3.27.3 QG (MATRIX)

Description

$[q_g]$ - Single-point constraint forces and determinate support forces matrix - q set.

Matrix Trailer

Number of columns	=	number of subcases in CASECC
Number of rows	=	g
Form	=	rectangular
Type	=	real single precision

2.3.27.4 PHIG (MATRIX)

Description

$[\Phi_g]$ - Eigenvector matrix giving eigenvectors (displacements) in the g set.

Matrix Trailer

Number of columns	=	number of eigenvalues found in READ
Number of rows	=	g
Form	=	rectangular
Type	=	real single precision

2.3.27.5 UBGV (MATRIX)

Description

$[u_g^b]$ - Displacement vector matrix for differential stiffness giving displacements in the g set.

Matrix Trailer

Number of columns = number of factors on a DSFACT bulk data card
Number of rows = g
Form = rectangular
Type = real single precision

2.3.27.6 QBG (MATRIX)

Description

$[q_g^b]$ - Single-point forces of constraint matrix for differential stiffness - g set.

Matrix Trailer

Number of columns = number of factors on a DSFACT bulk data card
Number of rows = g
Form = rectangular
Type = real single precision

2.3.27.7 BQG (MATRIX)

Description

$[q_g^b]$ - Single-point forces of constraint matrix for a buckling analysis problem - g set.

Matrix Trailer

Number of columns = number of buckling modes found in READ
Number of rows = g
Form = rectangular
Type = real single precision

2.3.27.8 DELTAUGV (MATRIX)

Description

$\{\delta u_g\}$ - Incremental displacement vector in piecewise linear analysis - g set.

Matrix Trailer

Number of columns = 1
Number of rows = g
Form = rectangular
Type = real single precision

2.3.27.9 DELTAPG (MATRIX)

Description

$\{\delta P_g\}$ - Incremental load vector in piecewise linear analysis - g set.

Matrix Trailer

Number of columns = 1
Number of rows = g
Form = rectangular
Type = real single precision

2.3.27.10 DELTAQG (MATRIX)

Description

$\{\delta q_g\}$ - Incremental vector of single-point forces of constraint in piecewise linear analysis - g set.

Matrix Trailer

Number of columns = 1
Number of rows = g
Form = rectangular
Type = real single precision

2.3.27.11 CPHIP (MATRIX)

Description

$[\Phi_p]$ - Complex eigenvectors in p set.

Matrix Trailer

Number of columns = number of eigenvalues found in CEAD
Number of rows = p
Form = rectangular
Type = complex single precision

2.3.27.12 QPC (MATRIX)

Description

$[q_p^c]$ - Complex single-point forces of constraint - p set.

Matrix Trailer

Number of columns = number of eigenvalues found in CEAD
Number of rows = p
Form = rectangular
Type = complex single precision

2.3.27.13 UPVC (MATRIX)

Description

$[u_p^c]$ - Frequency response solution vectors - p set.

Matrix Trailer

Number of columns = the product of the number of frequencies and number of loads
Number of rows = p
Form = rectangular
Type = complex single precision

2.3.27.14 UPV (MATRIX)

Description

$[u_p]$ - Transient solution vectors - p set.

Matrix Trailer

Number of columns = the number of output times multiplied by 3
Number of rows = p
Form = rectangular
Type = real single precision

2.3.27.15 QP (MATRIX)

Description

$[q_p]$ - Transient single-point forces of constraint - p set.

Matrix Trailer

Number of columns = the number of output times multiplied by 3.
Number of rows = p
Form = rectangular
Type = real single precision

2.3.28 Data Blocks Output From Module SDR2.

2.3.28.1 ØUGV1 (TABLE)

Description

Output displacement vector requests (g set, SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code } repeat
	2	I	Point type } for each
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } point

Notes

1. Records 1 and 2 are repeated for each vector to be output

2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch

3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase

4. Approach code = 1, 3, 7, or 10

5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.28.2 ØUBGV1 (TABLE)

Description

Output displacement vector requests (g set, SØRT1, real)

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code — repeat for each point
	2	I	Point type — repeat for each point
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) — repeat for each point

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }
3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }
4. Approach code = 4
5. Point type = { 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point }

Table Trailer

Words 1-6 contain no significant values.

2.3.28.3 ØUPV1 (TABLE)

Description

Output displacement vector requests (p set, SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1 = Displacement 10 = Velocity 11 = Acceleration
	3	I	0
	4	I	Subcase number
	5	R	Time
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code (repeat for each point)
	2	I	Point type (repeat for each point)
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) (repeat for each point)

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 6
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.28.4 ØUPVC1 (TABLE).

Description

Output displacement vector requests (p set, SØRT1, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1001 = Displacement 1010 = Velocity 1011 = Acceleration
	3	I	0
	4	I	Subcase number
	5	R	Frequency
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code (repeat for each point)
	2	I	Point type (repeat for each point)
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) (repeat for each point)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3) (repeat for each point)

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 5
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.28.5 ØPG1 (TABLE).

Description

Output load vector requests (g set, SØRT1, real)

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code } Repeat
	2	I	Point type } for each
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } point

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 1, 3, 7, or 10
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.28.6 ØPP1 (TABLE).

Description

Output load vector requests (p set, SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2
	3	I	0
	4	I	Subcase number
	5	R	Time
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code } repeat
	2	I	Point type } for each
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code =
   - 0 = x y output only
   - 1 = print
   - 4 = punch
   - 5 = print and punch

3. Format code =
   - 1 = real
   - 2 = real/imaginary
   - 3 = magnitude/phase

4. Approach code = 6

5. Point type =
   - 1 = grid point
   - 2 = scalar point
   - 3 = extra point
   - 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.28.7 ØPPC1 (TABLE).

Description

Output load vector requests (p set, SØRT1, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1002
	3	I	0
	4	I	Subcase number
	5	R	Frequency
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code (repeat for each point)
	2	I	Point type (repeat for each point)
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) (repeat for each point)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3) (repeat for each point)

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 5
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.28.8 ØQG1 (TABLE)

Description

Output forces of single-point constraint requests (g set, SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code } repeat
	2	I	Point type } for each
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } point

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }
3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }
4. Approach code = 1, 2, 3, 7, or 10
5. Point type = { 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point }

Table Trailer

Words 1-6 contain no significant values.

2.3.28.9 ØQBG1 (TABLE)

Description

Output forces of single-point constraint requests (g set, SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code } repeat
	2	I	Point type } for each
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } point

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }
3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }
4. Approach code = 4
5. Point type = { 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point }

Table Trailer

Words 1-6 contain no significant values.

2.3.28.10 ØBQG1 (TABLE).

Description

Output forces of single-point constraint requests (g set, SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3
	3	I	0
	4	I	Subcase number
	5	I	Mode number
	6	R	Eigenvalue
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code } repeat
	2	I	Point type } for each
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 8

5. Point type = { 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point }

Table Trailer

Words 1-6 contain no significant values.

2.3.28.11 ØQP1 (TABLE).

Description

Output forces of single-point constraint requests (p set, SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3
	3	I	0
	4	I	Subcase number
	5	R	Time
	6	I	0
	7	I	C
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code — repeat for each point
	2	I	Point type — repeat for each point
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) — repeat for each point

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 6
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.28.12 ØQPC1 (TABLE).

Description

Output forces of single-point constraint requests (p set, SØRT1, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1003
	3	I	0
	4	I	Subcase number
	5	R/I	Frequency or mode number
	6	I/R	0 or eigenvalue (real part)
	7	I/R	0 or eigenvalue (real part)
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code) repeat
	2	I	Point type) for
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)) each
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)) point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 5, or 9

5. Point type = { 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point }

Table Trailer

Words 1-6 contain no significant values.

2.3.28.13 ØPHIG (TABLE).

Description

Output eigenvector requests (g set, SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	7
	3	I	0
	4	I	Subcase number
	5	I	Mode number
	6	R	Eigenvalue
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code } repeat
	2	I	Point type } for each
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 2, or 8

5. Point type = { 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point }

Table Trailer

Words 1-6 contain no significant values.

2.3.28.14 ØCPHIP (TABLE).

Description

Output eigenvector requests (p set, SØRT1, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1007
	3	I	0
	4	I	Subcase number
	5	I	Mode number
	6	R	Eigenvalue (real part)
	7	R	Eigenvalue (imaginary part)
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code) repeat
	2	I	Point type) for
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)) each
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)) point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 9

5. Point type = { 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point }

Table Trailer

Words 1-6 contain no significant values.

2.3.28.15 ØES1 (TABLE).

Description

Output element stress requests (SØRT1, real).

Table Format

Record	Word	Type	Item	
0			Header record	
1	1	I	Device code + 10*approach code	
	2	I	5	
	3	I	Element type	
	4	I	Subcase number	
	5	I/R	Time, Load set ID, or mode number	
	6	R/I	Eigenvalue or 0	
	7	I	0	
	8	I	Load set ID or 0	
	9	I	Format code	
	10	I	Number of words per entry in next record = NWDS	
	11-50		Not defined	
	51-82	B	Title	
	83-114	B	Subtitle	
	115-146	B	Label	
2	1	I	10*element ID + device code	repeat for each element
	2-NWDS	Mixed	Element stress data See 2.3.51 for details	repeat for each element

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 1, 2, 3, 6, 7, or 10

Table Trailer

Words 1-6 contain no significant values.

2.3.28.16 ØESB1 (TABLE).

Description

Output element stress requests (SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	5
	3	I	Element type
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code } repeat
	2-NWDS	Mixed	Element stress data } for each
			See 2.3.51 for details } element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 4

Table Trailer

Words 1-6 contain no significant values.

2.3.28.17 ØBES1 (TABLE).

Description

Output element stress requests (SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	5
	3	I	Element type
	4	I	Subcase number
	5	I	Mode number
	6	R	Eigenvalue
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code } repeat
	2-NWDS	Mixed	Element stress data } for each
			See 2.3.51 for details } element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch

3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase

4. Approach code = 8

Table Trailer

Words 1-6 contain no significant values.

2.3.28.18 ØESC1 (TABLE).

Description

Output element stress requests (SØRT1, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1005
	3	I	Element type
	4	I	Subcase number
	5	R/I	Frequency or mode number
	6	I/R	0 or eigenvalue (real part)
	7	I/R	0 or eigenvalue (imaginary part)
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code } repeat
	2-NWDS	Mixed	Element stress data } for each
			See 2.3.51 for details } element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 5, or 9

Table Trailer

Words 1-6 contain no significant values.

2.3.28.19 ØEF1 (TABLE).

Description

Output element force requests (SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	4
	3	I	Element type
	4	I	Subcase number
	5	I/R	Time, load set ID, or mode number
	6	I/R	0 or eigenvalue
	7	I	0
	8	I	Load set ID or 0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code } repeat
	2-NWDS	Mixed	Element force data } for each
			See 2.3.52 for details } element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 1, 2, 3, 6, 7, or 10

Table Trailer

Words 1-6 contain no significant values.

2.3.28.20 ØEFBl (TA3LE).

Description

Output element force requests (SØRTl, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	4
	3	I	Element type
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code } repeat
	2-NWDS	Mixed	Element force data } for each
			See 2.3.52 for details } element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 4

Table Trailer

Words 1-6 contain no significant values.

2.3.28.21 ØBEF1 (TABLE).

Description

Output element force requests (SØRT1, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	4
	3	I	Element type
	4	I	Subcase number
	5	I	Mode number
	6	R	Eigenvalue
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code } repeat
	2-NWDS	Mixed	Element force data } for each
			See 2.3.52 for details } element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 8

Table Trailer

Words 1-6 contain no significant values.

2.3.28.22 ØEFC1 (TABLE).

Description

Output element force requests (SØRT1, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1004
	3	I	Element type
	4	I	Subcase number
	5	R/I	Frequency or mode number
	6	I/R	0 or eigenvalue (real part)
	7	I/R	0 or eigenvalue (imaginary part)
	8	I	Load set ID or 0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code) repeat
	2-NWDS	Mixed	Element force data) for each
			See 2.3.52 for details) element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch

3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase

4. Approach code = 5, or 9

Table Trailer

Words 1-6 contain no significant values.

2.3.28.23 PUGV1 (matrix - see note below)

Description

PUGV1 contains the translation components of UGV1 rotated to the basic coordinate system.

Note

The first four words of each logical record (column) contain identification data for the column. These words must be read prior to calling the appropriate unpacking routine.

Word 1 = subcase number

Word 2 = 1

Word 3 = static load set ID

Word 4 = 0

Matrix Trailer

Trailer is same as that for UGV1 except word 1 = 0, and word 6 = 0 (see section 2.3.36.1).

2.3.28.24 PUBGV1 (matrix - see note below)

Description

PUBGV1 contains the translation components of UBGV rotated to the basic coordinate system.

Note

The first four words of each logical record (column) contain identification data for the column. These words must be read prior to calling the appropriate unpacking routine.

Word 1 = subcase number

Word 2 = 1

Word 3 = static load set ID

Word 4 = 0

Matrix Trailer

Trailer is same as that for UBGV with word 1 = 0, and word 6 = 0.

2.3.28.25 PPHIG (matrix - see note below)

Description

PPHIG contains the translation components of PHIG rotated to the basic coordinate system.

Note

The first four words of each logical record (column) contain identification data for the column. These words must be read prior to calling the appropriate unpacking routine.

Word 1 = subcase number

Word 2 = 2

Word 3 = mode number

Word 4 = eigenvalue (λ)

Matrix Trailer

Trailer is same as that for PHIG with word 1 = 0, and word 6 = 0, (see section 2.3.27.4).

2.3.28.26 PUGV (matrix - see note below)

Description

PUGV contains the translation components of UPV (excluding extra points) rotated to the basic coordinate system.

Note

The first four words of each logical record (column) contain identification data for the column. These words must be read prior to calling the appropriate unpacking routine.

Word 1 = subcase number

Word 2 = 3

Word 3 = 0

Word 4 = time

Matrix Trailer

Trailer is same as that for UGV with word 1 = 0, and word 6 = 0, (see section 2.3.27.1).

2.3.29 Data Blocks Output From Module DPD

2.3.29.1 GPLD (TABLE)

Description

Grid Point List Dynamics.

One logical record which contains a list of all grid points, scalar points and extra points in the model in internal sort.

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	ID for first point
	⋮		⋮
	n	I	ID for n^{th} point
2			End-of-file

Table Trailer

Word 1 = number of grid points + number of scalar points + number of extra points.

Word 2-6 = zero.

2.3.29.2 SILD (TABLE)

Description

Scalar Index List Dynamics.

Two logical records. First logical record contains scalar index values in the p-displacement set for each point in the dynamics model (internal order). These values are defined as follows:

$$SILD_1 = 1$$

$$SILD_{i+1} = \begin{cases} {}^{T}LD_i + 6 & \text{if i corresponds to a grid point} \\ {}^{'}D_i + 1 & \text{if i corresponds to a scalar or an extra point} \end{cases}$$

The second logical record contains an equivalence between scalar index values in the g-displacement set and scalar index values in the p-displacement set.

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Scalar index for first point
	⋮	⋮	⋮
	n	I	Scalar index for n^{th} point
2	1, 2	I	SIL value, SILD value
	⋮	⋮	⋮
	2m-1,2m	I	SIL value, SILD value
3			End-of-file

Table Trailer

Word 1 = degrees of freedom in the p-displacement set (LUSETD).

Word 2 = number of extra points.

Word 3-6 = zero.

2.3.29.3 USETD (TABLE)

Description

Displacement set definitions table dynamics.

USETD contains one logical record. Each word corresponds to each degree of freedom in the p-displacement set (in internal order) and contains ones in specified bit positions indicating the displacement sets to which the point belongs.

Table Format

Record	Word	Type	Item
0			Header record
1	1	L	Mask for first degree of freedom
	⋮	⋮	⋮
	n	L	Mask for n^{th} degree of freedom
2			End-of-file

Notes

Bit positions for the various displacement sets are defined as follows:

d	f_e	n_e	p	e	s_b	s_g	ℓ	a	f	n	g	r	o	s	m
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

Table Trailer

Word 1 = degrees of freedom in the p-displacement set (LUSETD).

Word 2 = number of extra points.

Word 3 = zero.

Word 4 = logical "or" of all USETD masks.

Word 5 = zero.

Word 6 = zero.

2.3.29.4 EED (TABLE)

Description

Eigenvalue Extraction Data.

The EED contains one logical record for each eigenvalue extraction card type (EIGB, EIGC, EIGP, EIGR). Each logical record contains data from all cards of a given type.

Table Format

Record	Word	Item
0		Header record
1		EIGB data (if EIGB cards in bulk data)
2		EIGC data (if EIGC cards in bulk data)
3		EIGP data (if EIGP cards in bulk data)
4		EIGR data (if EIGR cards in bulk data)
5		End-of-file

Detailed format for EIGB data:

Word	Type	Item	
1-3	I	107, 1, 0	repeated for each EIGB card in bulk data
4	I	Set ID	
5-6	B	Method	
7-8	R	F_1, F_2	
9-11	I	N_e, N_d, N_z	
12	R	E	
13-14	B	Norm	
15	I	If norm = "PØINT", SIL value in a-set of normalization point	
16-21		Not defined	

Detailed format for EIGC card:

Word	Type	Item	
1-3	I	207, 2, 0	repeated for each EIGC card in bulk data
4	I	Set ID	
5-6	B	Method	
7-8	B	Norm	
9	I	If Norm = "PØINT", SIL value in analysis set of normalization point	
10		Not defined	
11	R	E	
12-13		Not defined	
14-15	R	α_a, ω_a	repeated for each region definition
16-17	R	α_b, ω_b	
18	R	ℓ	
19-20	I	N_e, N_d	
21		Not defined	
14+8k-21+8k	I	-1 (k = number of regions)	

Detailed format for EIGP card:

Word	Type	Item	
1-3	I	257, 4, 0	
5	I	Set ID	repeated for each EIGP card in bulk data
6-7	R	α, ω	
8	I	M	

Detailed format for EIGR card:

Word	Type	Item	
1-3	I	307, 3, 0	repeated for each EIGR card in bulk data
4	I	Set ID	
5-6	B	Method	
7-8	R	F_1, F_2	
9-11	I	N_e, N_d, N_z	
12	R	E	
13-14	B	Norm	
15	I	If norm = "PØINT", SIL value in a-set of normalization point	
16-21		Not defined	

Table Trailer

Word 1 =

bit 17 = 1 if EIGB record exists
18 = 1 if EIGC record exists
19 = 1 if EIGP record exists
20 = 1 if EIGR record exists
other bits = 0

Word 2-6 = zero.

2.3.29.5 EQDYN (TABLE)

Description

Equivalence between external points and scalar index values - dynamics.

EQDYN contains two logical records. The first record contains pairs of external point numbers and scalar index values in the p-displacement set for the points in external order. The second record is essentially the same as the first except that the type of point (grid, scalar, extra) is coded in the second word of the pair.

Table Format

Record	Word	Type	Item
0			Header record
1	1,2	I	ID for first point, scalar index for first point
	⋮	⋮	⋮
	2n-1,2n	I	ID for n^{th} point, scalar index for n^{th} point
2	1,2	I	ID for first point, 10*scalar index + type
	⋮	⋮	⋮
	2n-1,2n	I	ID for n^{th} point, 10*scalar index + type
3			End-of-file

Note

Type = {1 for grid point; 2 for scalar point; 3 for extra point}

Table Trailer

Word 1 = number of grid points + number of scalar points + number of extra points in dynamics model.

Word 2 = number of extra points.

Word 3-6 = zero.

2.3.29.6 TFPØØL (TABLE).

Description

Transfer Function Pool.

The TFPØØL data block contains one logical record for each transfer function set defined in the bulk data on a TF bulk data card. Point and component values are converted to row and column numbers in the p-displacement set.

Table Format

Record	Word	Type	Item	
0			Header record	
1	1	I	Set ID	
	2	I	65536*column number + row number	repeated for each set of non-zero terms
	3-5	R	Coefficients	
n			Same format as first record	
n+1			End-of-file	

Table Trailer

Word 1 = number of transfer function sets.

Word 2-6 = zero.

2.3.29.7 DLT (TABLE).

Description

Dynamic Loads Table.

The header record of the DLT contains a summary of all dynamic load sets defined in the problem. The first record of the DLT contains all DLØAD cards (if DLØAD cards have been input). Each succeeding record contains all data for one dynamic load set.

Table Format

Record	Word	Type	Item	
0	1-2	B	Data block name	
	3	I	m = number of DLØAD set ID's	
	4-3+m	I	Set ID's on DLØAD cards	
	4+m-3+m+n	I	Set ID's on RLØAD1, 2 and TLØAD1, 2 cards	
1	1	I	Set ID	
	2	R	Scale factor	repeated for each DLØAD card
	3-4	R,I	Scale factor, set ID	
	:			
	4+ℓ,5+ℓ	I	-1, -1	

Record	Word	Type	Item
k	1	I	Dynamic load card type
	2	I	= 0 no time delays ≠ 0 time delays
	3-8		See Notes
	9	I	SIL number ⎫ repeated for each
	10-12	R	A, τ, θ ⎭ dynamic load set
n+2			End-of-file

Notes

1. If no DLØAD cards have been input, the third word of the header record is zero and the DLØAD record does not exist. Therefore, record 1 of the DLT corresponds to the load set defined in word 4 of the header record.

2. DLØAD-set ID's are in sort by set ID. In record 1, set ID's within a DLØAD card are in sort.

3. Within other records, data is in sort by SIL number.

4. Formats by dynamic load card type are as follows:

1 = RLØAD1

Word	Type	Item
3	I	Table ID for C(f)
4	I	Table ID for D(f)
5-8		Not defined

2 = RLØAD2

Word	Type	Item
3	I	Table ID for B(f)
4	I	Table ID for ϕ(f)
5-8		Not defined

3 = TLØAD1

Word	Type	Item
3	I	Table ID for F(t)
4-8		Not defined

4 = TLØAD2

Word	Type	Item
3-4	R	T_{K1}, T_{K2}
5-6	R	W_K, Φ_K
7-8	R	n_K, α_K

Table Trailer

Word 1 = GINØ file name of DLT.
Word 2-6 = undefined.

2.3.29.8 PSDL (TABLE)

Description

Power Spectral Density List.

The first logical record of the PSDL contains RANDPS data. Subsequent logical records contain RANDT data, one set per logical record. Each RANDT logical record contains a sorted list of unique time lags defined in the set.

Table Format

Record	Word	Type	Item
0	1,2	B	Data block name
	3	I	RANDT set ID_1
	⋮		⋮
	2+n	I	RANDT set ID_n
1	1	I	RANDPS set ID) repeated
	2	I	Load set ID (for each
	3	I	Load set ID } RANDPS
	4,5	R	Complex number (card in
	6	I	Table ID) bulk data
2	1-m	R	Time lags
⋮			
n+1			Same format as record 2 Data belongs to RANDT set ID_n
n+2			End-of-file

Notes

1. RANDPS cards must be present for data block to exist. Therefore, record one always contains RANDPS data.

2. If no RANDT1 or RANDT2 cards are present in the bulk data, the header record will contain exactly two words and record two will be an end-of-file.

Table Trailer

Word 1 = { number of RANDT sets, or 65535 if no RANDT sets exist.

Word 2-6 = zero.

2.3.29.9 FRL (TABLE)

Description

Frequency Response List.

The FRL contains one logical record for each different frequency set defined in the bulk data. Each record contains a sorted list of the unique frequencies defined in the set.

Table Format

Record	Word	Type	Item
0	1,2	B	Data block name
	3	I	Set ID_1
	⋮		⋮
	2+n	I	Set ID_n
1	1-m	R	Frequencies belonging to set ID_1
⋮			⋮
n	1-k	R	Frequencies belonging to set ID_n
n+1			End-of-file

Table Trailer

Word 1 = number of frequency sets.

Word 2-6 = zero.

2.3.29.10 NLFT (TABLE)

Description

Non-Linear Forcing Table.

The header record of the NLFT contains a sorted list of set identification numbers for all NØLIN sets defined in the bulk data. Each logical record of the NLFT contains all data for a single set. Point and component numbers on the NØLIN cards are converted to scalar index values in both the d- and e-displacement sets.

Table Format

Record	Word	Type	Item
0	1,2	B	Data block name
	3	I	Set ID_1
	⋮		⋮
	2+n	I	Set ID_n

Record	Word	Type	Item	
1	1	I	Type of nonlinear load (1≤type≤4)	repeated for each NØLIN card in set
	2	I	SIL value in d-set	
	3	I	SIL value in e-set	
	4	R	Scale factor	
	5	I	SIL value in d-set	
	6	I	SIL value in e-set	
	7	type = 1 =	Table ID (integer)	
		type = 2 =	SIL value in d-set (integer)	
		type = 3 =	Scale factor (real)	
		type = 4 =	Scale factor (real)	
	8	type = 1 =	Not defined	
		type = 2 =	SIL value in e-set (integer)	
		type = 3 =	Not defined	
		type = 4 =	Not defined	
⋮				
n			Same format as record 1. Data belongs to set ID_n.	
n+1			End-of-file	

Note

Within each record, the data is sorted on word 2 of each 8-word entry in the record.

Table Trailer

Word 1 = number of NØLIN sets.

Word 2-6 = zero.

2.3.29.11 TRL (TABLE).

Description

Transient Response List.

The header record of the TRL contains a list of all transient initial condition set identifications in the bulk data. Subsequent logical records contain TIC data for each set (one set per logical record). If TSTEP cards are present in the bulk data, this data follows the TIC data, one logical record for each TSTEP set.

Table Format

Record	Word	Type	Item
0	1,2	B	Data block name
	3	I	Number of TIC sets
	4	I	Set ID_1
	⋮		⋮
	3+n	I	Set ID_n
	4+n	I	Degrees of freedom in the d-displacement set
1	1	I	SIL value in d-set } repeated for each initial
	2,3	R	U_o, V_o } condition in set
⋮			
n			Same format as record 1 Data belongs to set ID_n
n+1	1	I	TSTEP set ID
	2	I	N } repeated for
	3	R	Δt } each interval
	4	I	NO } in set
⋮			
n+m			Same format as record n+1
n+m+1			End-of-file

Notes

1. Data within each TIC record is sorted on word 1 of each 3-word entry.
2. If word 3 of the header record = 0, then the first logical record of the TRL contains TSTEP data.
3. If TSTEP data is not present in the bulk data, and end-of-file will follow the last TIC record.

Table Trailer

Word 1 = number of TIC sets.

Word 2 = number of TSTEP sets.

Word 3-6 = zero.

2.3.30 Data Blocks Output From Module READ

2.3.30.1 LAMA (TABLE)

Description

λ_a - Real Eigenvalue Table

Table Format

Record	Word	Type	Item
0			Header record
1			OFP ID record
	1	I	21
	2	I	9
	3	I	0
	4-9	I	0
	10	I	7
	11-50		Not defined
	51-146	A	Title, subtitle, and label from /OUTPUT/
2			OFP data record
	1	I	Mode number
	2	I	Extraction order
	3	R	λ - eigenvalue
	4	R	$f = \sqrt{\lvert\lambda\rvert}$
	5	R	$w = f/2\pi$
	6	R	Generalized mass
	7	R	Generalized stiffness
3			End-of-file

Notes

1. The seven data words in record 2 repeat for each eigenvalue found in READ.

Table Trailer

Non-zero trailer

2.3.30.2 PHIA (MATRIX)

Description

$[\Phi_a]$ - Eigenvectors matrix giving the eigenvectors (displacements) in the a set.

Matrix Trailer

Number of columns = number of eigenvalues found in READ
Number of rows = a
Form = rectangular
Type = real single precision

2.3.30.3 MI (MATRIX)

Description

$[m_i]$ - Modal Mass Matrix

Matrix Trailer

Number of columns = number of eigenvalues found in READ
Number of rows = number of eigenvalues found in READ
Form = general
Type = real single precision

2.3.30.4 ØEIGS (TABLE)

Description

Real Eigenvalue Summary Table

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	21
	2	I	9
	3	I	2 If Inverse Power Method 1 If Determinant Method 4 If Givens Method
	4	I	0
	5	I	0
	6	I	0
	7	I	0
	8	I	0
	9	I	0
	10	I	0

Words 11-17 depend on the method used.

Determinant Method:

11	I	Number of eigenvalues extracted
12	I	Number of passes through starting points
13	I	Number of criteria changes
14	I	Number of starting point moves
15	I	Number of triangular decompositions
16	I	Number of failures to iterate to a root
17	I	Reason for termination 1 - All requested roots formed 2 - Out of region predictions from every starting point 3 - Insufficient time to extract another root 4 - Everywhere singular matrix

Record	Word	Type	Item
			Inverse Power Method:
	11	I	Number of eigenvalues extracted
	12	I	Number of starting points used
	13	I	Number of starting points moved
	14	I	Number of triangular decompositions
	15	I	Number of vector iterations
	16	I	Dummy
	17	I	Reasons for termination 1 - 2 Singularities encountered in a row 2 - 4 Shifts while tracking one root 3 - Regions completed 4 -3* Number of estimated roots were found 5 - All roots of problem found 6 - Number desired roots found 7 - λ outside maximum range 8 - Insufficient time to extract another root 9 - 200 iterations and 1 shift point move before locating a root
			Givens Method:
	11	I	Number of eigenvalues extracted
	12	I	Number of eigenvectors computed
	13	I	Number of failures to converge to an eigenvalue
	14	I	Number of failures to converge to an eigenvector
	15	I	Dummy
	16	I	Dummy
	17	I	Reason for termination 0 - Normal termination 3 - Insufficient time to evaluate eigenvectors
	18	R	Value of off-diagonal element of modal mass matrix having largest magnitude (zero where not applicable)
	19	I	Column of 18 in MI
	20	I	Row of 18 in MI
	21	I	Number of off-diagonal elements of modal mass matrix that fail to meet error criterion
	22-50		Not used
	51-146	B	Title, subtitle, label
			Records 2 and 3 exist only when the Determinant Method is used.
2	1	I	21
	2	I	9
	3	I	3
	4	I	0
	5	I	0
	6	I	0
	7	I	0
	8	I	0
	9	I	0
	10	I	6
	11-50		Not used
	51-146	B	Title, subtitle, label
3	1	I	Starting point ID
	2	R	λ - Starting point
	3	R	$\omega = \sqrt{\lambda}$ - Starting point
	4	R	$f = \omega/2\pi$ - Starting point
	5	R	Determinant at λ

Words 1-6 are repeated for each starting point

Record	Word	Type	Item
	6	R	Scale factor (power of 10) of determinant
4			End-of-file

Table Trailer

Nonzero

2.3.31 Data Blocks Output From Module DSMG1

2.3.31.1 KDGG (MATRIX)

Description

$[K^{d}_{gg}]$ - Partition of differential stiffness matrix - g set.

Matrix Trailer

Number of columns	=	g
Number of rows	=	g
Form	=	symmetric
Type	=	real double precision

2.3.32 Data Blocks Output From Module SMP2

2.3.32.1 KDAA (MATRIX)

Description

$[K^{d}_{aa}]$ - Partition of differential stiffness matrix - a set.

Matrix Trailer

Number of columns	=	a
Number of rows	=	a
Form	=	symmetric
Type	=	real double precision

2.3.33 Data Blocks Output From Module DSMG2

2.3.33.1 KBLL (MATRIX)

Description

$[K^b_{\ell\ell}]$ - Partition of the stiffness matrix of the first order approximation to large displacements - ℓ set.

Matrix Trailer

Number of columns = ℓ
Number of rows = ℓ
Form = symmetric
Type = real double precision

2.3.33.2 KBFS (MATRIX)

Description

$[K^b_{fs}]$ - Partition of the stiffness matrix of the first order approximation to large displacements.

Matrix Trailer

Number of columns = s
Number of rows = f
Form = rectangular
Type = real double precision

2.3.33.3 KBSS (MATRIX)

Description

$[K^b_{ss}]$ - Partition of the stiffness matrix of the first order approximation to large displacements - s set.

Matrix Trailer

Number of columns = s
Number of rows = s
Form = symmetric
Type = real double precision

2.3.33.4 PBL (MATRIX)

Description

$\left\{P_{\ell}^{b}\right\}$ - Partition of the load vector of the first order approximation to the large displacements - ℓ set.

Matrix Trailer

Number of columns = 1
Number of rows = ℓ
Form = rectangular
Type = real single precision

2.3.33.5 PBS (MATRIX)

Description

$\left\{P_{s}^{b}\right\}$ - Partition of the load vector of the first order approximation to the large displacement problem - s set.

Matrix Trailer

Number of columns = 1
Number of rows = s
Form = rectangular
Type = real single precision

2.3.33.6 YBS (MATRIX)

Description

$\left\{Y_{s}^{b}\right\}$ - Partition of the constrained displacement vector of the first order approximation to the large displacement vector - s set.

Matrix Trailer

Number of columns = 1
Number of rows = s
Form = rectangular
Type = real single precision

2.3.33.7 URØØV (MATRIX)

Description

$\left\{u_{o}^{ob}\right\}$ - Partition of the displacement vector of the first order approximation to the large displacement problem - o set.

Matrix Trailer

Number of columns = 1
Number of rows = o
Form = rectangular
Type = real single precision

2.3.34 Data Blocks Output From Module PLA1.

2.3.34.1 KGGXL (MATRIX).

Description

$[K_{gg}^{x\ell}]$ - Stiffness matrix of linear elements exclusive of general elements - g set.

Matrix Trailer

Number of columns	=	g
Number of rows	=	g
Form	=	symmetric
Type	=	real double precision

2.3.34.2 ESTL (TABLE).

Description

Element Summary Table for Linear Elements.

The ESTL contains data copied from the EST data block. An element's EST data resides in the ESTL only if it is a linear element of the model.

Table Format

Same format as the EST data block output from module TA1.

Table Trailer

Word 1 = number of element entries in ESTL.

Words 2-6 = zero.

2.3.34.3 ESTNL (TABLE).

Description

Element Summary Table for Non-Linear Elements.

The ESTNL, used only in the Piecewise Linear Analysis Rigid Format, is constructed from the Element Summary Table (EST). It contains one logical record for each element type for which at least one element of that type is non-linear (an element is defined to be non-linear if its modulus of elasticity is defined as the first derivative of a stress-strain tabular function input on a TABLES1 bulk data card) and for which a request for stress output is found. The construction of the ESTNL is as follows: the EST data block is read and each element is tested for possible non-linearity. If the element is non-linear and the user has requested element stress data to be output, its element data is copied onto the ESTNL data block and then initial stress data is appended. The only elements admissible to the ESTNL are: RØD, TUBE, CØNRØD, BAR, TRMEM, TRIA1, TRIA2, QDMEM, QUAD1, QUAD2.

Table Format

Record	Word	Item	
0		Header record	
1	1	Element type (integer)	repeated for each admissible element type
	2 to N+1	Element EST data (repeated for each non-linear element)	
	N+2 to N+M+1	Element stress data (repeated for each non-linear element)	

Notes

1. N is the number of words in the EST data section.
   M is the number of stress words appended.

2. The number of records in the ESTNL corresponds to the number of separate admissible element types for which at least one element is non-linear.

Table Trailer

Word 1 = total number of elements in the ESTNL.

Words 2-6 = zero.

Detailed ESTNL Formats

RØD, CØNRØD:

Word	Item
1-17	EST data
18	ε_0^*, previous strain value once removed, initially zero
19	ε^*, previous strain value, initially zero
20	E^*, the previously calculated modulus of elasticity, initially the value of E given a MAT1 card.
21	T^*, the previously calculated torsional moment, initially zero

TUBE:

Word	Item
1-16	EST data
17-20	Same as words 18-21 for the RØD.

BAR:

Word	Item
1-42	EST data
43	ε_0^*, previous strain value once removed, initially zero
44	ε^*, previous strain value, initially zero
45	E^*, the previously calculated modulus of elasticity, initially the value of E given on a MAT1 card
46	V_1^*
47	V_2^*
48	T^* } The previous element forces, initially zero
49	M_{1a}^*
50	M_{2a}^*

TRMEM:

Word	Item
1-21	EST data
22	ε_0^*, previous strain value once removed, initially zero
23	ε^*, previous strain value, initially zero
24	E^*, the previously calculated modulus of elasticity, initially the value of E given on a MAT1 card
25	σ_x^*
26	σ_y^* } The current membrane stresses, initially zero
27	σ_{xy}^*

TRIA1:

Word	Item
1-27	EST data
28-33	Same as words 22-27 for the TRMEM

Word	Item
34	M_x^*
35	M_y^*
36	M_{xx}^*
37	V_x^*
38	V_y^*

The previous element forces, initially zero

TRIA2:

Word	Item
1-27	Same as words 1-27 for the TRMEM
28-32	Same as words 34-38 for the TRIA1

QDMEM:

Word	Item
1-26	EST data
27-32	Same as words 22-27 for the TRMEM

QUAD1:

Word	Item
1-32	EST data
33-38	Same as words 22-27 for the TRMEM
39-43	Same as words 34-38 for the TRIA1

QUAD2:

Word	Item
1-26	EST data
27-32	Same as words 22-27 for the TRMEM
33-37	Same as words 34-38 for the TRIA1

2.3.34.4 ECPTNL (TABLE).

Description

Element Connection and Properties Table for Non-Linear Elements.

The ECPTNL, used only in the Piecewise Linear Analysis Rigid Format, is constructed from the ECPT data block. The ECPTNL contains one logical record for each grid point or scalar point of the model. Each logical record contains Element Summary Table (EST) data plus initial element stress data appended to this data for each non-linear element connected to the pivot point. (An element is defined to be non-linear if its modulus of elasticity is defined as the first derivative of a stress-strain tabular function input on a TABLES1 card). The only elements admissible to the ECPTNL are: RØD, TUBE, CØNRØD, BAR, TRMEM, TRIA1, TRIA2, QDMEM, QUAD1, QUAD2.

Table Format

Record	Word	Item	
0		Header record	
1	1	SIL number for "pivot" grid or scalar point (integer)	repeated for each grid or scalar point in the model
	2	Element type (integer)	repeated for each non-linear element connected to the pivot
	3 to N+2	Element EST data	
	N+3 to N+M+2	Element stress data	
n+1		End-of-file	

Notes

1. N is the number of words in the EST data section.
   M is the number of stress words appended. The number of stress words appended in generating the ECPTNL data block is not the same as in generating the ESTNL data block.

2. n is the total number of grid and scalar points in the model.

3. If all elements connected to a pivot point are linear, then the record contains only one word, the pivot point set negative.

Table Trailer

Word 1 = total number of element entries in the ECPTNL.

Words 2-6 = zero.

Detailed ECPTNL Formats

RØD, CØNRØD:

Word	Item
1-20	Same as ESTNL data. Note word 21 of the ESTNL is not present in the ECPTNL data for the RØD, CØNRØD.

TUBE:

Word	Item
1-19	Same as ESTNL data. Note word 20 of the ESTNL is not present in the ECPTNL data for the TUBE.

BAR:

Word	Item
1-45	Same as ESTNL data. Note words 46-50 of the ESTNL are not present in the ECPTNL data for the BAR.

TRMEM:

Word	Item
1-27	Same as ESTNL data.

TRIA1:

Word	Item
1-33	Same as ESTNL data. Note words 34-38 of the ESTNL are not present in the ECPTNL data for the TRIA1.

TRIA2:

Word	Item
1-27	Same as ESTNL data. Note words 28-32 of the ESTNL are not present in the ECPTNL data for the TRIA2.

QDMEM:

Word	Item
1-32	Same as ESTNL data.

QUAD1:

Word	Item
1-38	Same as ESTNL data. Note words 39-43 of the ESTNL are not present in the ECPTNL data for the QUAD1.

QUAD2:

Word	Item
1-32	Same as ESTNL data. Note words 33-37 of the ESTNL are not present in the ECPTNL data for the QUAD2.

2.3.35 Data Blocks Output From Module ADD

2.3.35.1 KGGSUM (MATRIX)

Description

Sum of $[K_{gg}^{\ell}]$ and $[K_{gg}^{n\ell}]$.

Used only in the Piecewise Linear Analysis Rigid Format and is equivalent to $[K_{gg}]$.

Matrix Trailer

Number of columns = g
Number of rows = g
Form = symmetric
Type = real double precision

2.3.35.2 PG (MATRIX)

Description

$\left\{P_g\right\}$ - Incremental load vector used in Piecewise Linear Analysis.

Matrix Trailer

Number of columns = 1
Number of rows = g
Form = rectangular
Type = real single precision

2.3.35.3 KDAAM (MATRIX)

Description

$[K_{aa}^{dm}]$ - The negative of $[K_{aa}^{d}]$ (see section 2.3.32).

Used only in the Buckling Analysis Rigid Format.

Matrix Trailer

Number of columns = a
Number of rows = a
Form = symmetric
Type = real double precision

2.3.36 Data Blocks Output From Module PLA2

2.3.36.1 UGV1 (MATRIX)

Description

$[u_g^1]$ - Matrix of successive sums of incremental displacement vectors - g set. Used only in the Piecewise Linear Analysis Rigid Format.

Matrix Trailer

Number of columns	=	number of factors on a PLFACT bulk data card
Number of rows	=	g
Form	=	rectangular
Type	=	real single precision

2.3.36.2 PGV1 (MATRIX)

Description

$[P_g^1]$ - Matrix of successive sums of incremental load vectors - g set. Used only in the Piecewise Linear Analysis Rigid Format.

Matrix Trailer

Number of columns	=	number of factors on a PLFACT bulk data card
Number of rows	=	g
Form	=	rectangular
Type	=	real single precision

2.3.36.3 QG1 (MATRIX)

Description

$[q_g^1]$ - Matrix of successive sums of incremental vectors of single point constraint forces - g set. Used in the Piecewise Linear Analysis Rigid Format only.

Matrix Trailer

Number of columns	=	number of factors on a PLFACT bulk data card
Number of rows	=	g
Form	=	symmetric
Type	=	real single precision

2.3.37 Data Blocks Output From Module PLA3.

2.3.37.1 ØNLES (TABLE).

Description

Output table for nonlinear element stresses.

Format

Same format as ØES1 table output from module SDR2.

Note

ØNLES is written in subroutine PLA32 of module PLA3.

Table Trailer

Word 1 = total number of element entries in ØNLES.

Word 2-6 = zero.

2.3.37.2 ESTNL1 (TABLE).

Description

Element summary table for nonlinear elements - updated.

Used only in the Piecewise Linear Analysis Rigid Format, the ESTNL1 data block is the same as the ESTNL data block except that the appended stress information is updated. See data block description for ESTNL for further details.

Table Format

Same format as the ESTNL data block.

Table Trailer

Word 1 = number of element entries in ESTNL1.

Word 2-6 = zero.

2.3.38 Data Blocks Output From Module PLA4.

2.3.38.1 KGGNL (MATRIX).

Description

$[K_{gg}^{n\ell}]$ - Stiffness matrix of nonlinear elements - g set.

Used only in the Piecewise Linear Analysis Rigid Format.

Matrix Trailer

Number of columns	=	g
Number of rows	=	g
Form	=	symmetric
Type	=	real double precision

2.3.38.2 ECPTNL1 (TABLE).

Description

Element Connection and Properties Table for Non-Linear Elements - undated.

Used only in the Piecewise Linear Analysis Rigid Format, the ECPTNL1 data block is the same as the ECPTNL data block except that the appended stress information is updated. See description for ECPTNL for further details.

Table Format

Same format as the ECPTNL data block.

Table Trailer

Word 1 = total number of element entries in ECPTNL1.

Word 2-6 = zero.

2.3.39 Data Blocks Output From Module CASE.

2.3.39.1 CASEXX (TABLE).

Description

Case Control data table for dynamics problems.

Table Format

The format of the records is exactly like CASECC, (see section 2.3.1.1) with dynamic looping records deleted.

Table Trailer

Word 1 = number of records in CASEXX.

Word 2-6 = zero.

2.3.40 Data Blocks Output From Module MTRXIN

2.3.40.1 K2PP (MATRIX)

Description

$[K^{2}_{pp}]$ - Direct input stiffness matrix - p set.

Matrix Trailer

Number of columns	=	p
Number of rows	=	p
Form	=	square
Type	=	depends on input

2.3.40.2 M2PP (MATRIX)

Description

$[M^{2}_{pp}]$ - Direct input mass matrix - p set.

Matrix Trailer

Number of columns	=	p
Number of rows	=	p
Form	=	square
Type	=	depends on input

2.3.40.3 B2PP (MATRIX)

Description

$[B^{2}_{pp}]$ - Direct input damping matrix - p set.

Matrix Trailer

Number of columns	=	p
Number of rows	=	p
Form	=	square
Type	=	depends on input

2.3.41 Data Blocks Output From Module GKAD

2.3.41.1 KDD (MATRIX)

Description

$[K_{dd}]$ - Dynamic stiffness matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = complex double precision
- frequency response/complex eigenvalue
= real double precision
- transient

2.3.41.2 BDD (MATRIX)

Description

$[B_{dd}]$ - Dynamic damping matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = complex double precision
- frequency response/complex eignevalue
= real double precision
- transient

2.3.41.3 MDD (MATRIX)

Description

$[M_{dd}]$ - Dynamic mass matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = complex double precision
- frequency response/complex eigenvalue
= real double precision
- transient

2.3.41.4 GMD (MATRIX)

Description

$[G_m^d]$ - Multipoint constraint transformation matrix - dynamics.

Matrix Trailer

Number of columns = d
Number of rows = m
Form = rectangular
Type = complex double precision

2.3.41.5 GØD (MATRIX)

Description

$[G_o^d]$ - Omitted coordinate transformation matrix - dynamics.

Matrix Trailer

Number of columns = d
Number of rows = o
Form = rectangular
Type = complex double precision

2.3.41.6 K2DD(MATRIX)

Description

$[K_{dd}^2]$ - Direct input stiffness matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = complex double precision

2.3.41.7 M2DD (MATRIX)

Description

$[M_{dd}^2]$ - Direct input mass matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = complex double precision

2.3.41.8 B2DD (MATRIX)

Description

$[B^2_{dd}]$ - Direct input damping matrix - d set.

Matrix Trailer

Number of columns	=	d
Number of rows	=	d
Form	=	square
Type	=	complex double precision

2.3.42 Data Blocks Output From Module CEAD

2.3.42.1 PHID (MATRIX)

Description

$[\Phi_d]$ - Complex eigenvectors in the d set.

Matrix Trailer

Number of columns	=	number of eigenvalues found in CEAD
Number of rows	=	d
Form	=	rectangular
Type	=	complex single precision

2.3.42.2 CLAMA (TABLE)

Description

λ - Complex eigenvalue table.

Table Format

Record	Word	Type	Item
0			Header record
1			ØFP ID record
	1	I	90
	2	I	1006
	3-9	I	0
	10	I	6
	11-50		Not defined
	51-146	B	Title, subtitle, and label from /ØUTPUT/
2			ØFP data record
	1	I	Mode number
	2	I	Extraction order
	3	R	Real part of eigenvalue
	4	R	Imaginary part of eigenvalue
	5	R	$\lvert Im\ (\lambda)\rvert/2\Pi$
	6	R	$-2*Re\ (\lambda)/\lvert Im\ (\lambda)\rvert$
	Note: The 6 data words are repeated in record 2 for each eigenvalue found in CEAD.		
3			End-of-file

Table Trailer

Word 1	=	1006
Word 2	=	0
Word 3	=	0
Word 4	=	0
Word 5	=	6
Word 6	=	0

2.3.42.3 ØCEIGS (TABLE).

Description

Complex eigenvalue summary table.

Table Format

Record	Word	Type	Item
0		B	Header record
1	1	I	0
	2	I	1009
	3	I	1 if determinant 2 if inverse power
	4-10	I	0
	11	I	Number of eigenvalues extracted
	12-18		Depend on the method used
	Determinant		
	12	I	Number of passes through starting points
	13	I	Number of criteria changes
	14	I	Number of starting point moves
	15	I	Number of decompositions
	16	I	Number of failures to iterate to a root
	17	I	Number of predictions outside the region
	18	I	Reason for termination 1 - all requested roots found 2 - out of region prediction from every starting point 3 - insufficient time to extract another root 4 - everywhere singular matrix
	Inverse Power		
	12-18		Identical to words 12-18 for Inverse Power Method section of the ØEIGS data block output from the READ module (see section 2.3.30.4).
	19-50		Not defined
	51-146	B	Title, subtitle, label

Record 1 will be repeated for each region for Inverse Power.
Records 2 + 3 exist only when METHØD = DETM.

2	1	I	0
	2	I	1009
	3	I	3
	4-9	I	0
	10	I	6
	11-50		Not defined
	51-146	B	Title, subtitle, label

Record	Word	Type	Item
3	1	I	Starting point number in region
	2	R	Real part of starting point
	3	R	Imaginary part of starting point
	4	R	Magnitude of starting point
	5	R	Phase of starting point
	6	I	Scale factor (power of 10) of magnitude
	Words 1-6 are repeated for each starting point in each region.		
4			End-of-file

Table Trailer

Non-zero.

2.3.42.4 PHIH (MATRIX)

Description

$[\Phi_h]$ - Complex eigenvectors in the h set.

Matrix Trailer

Number of columns = number of eigenvalues found in CEAD
Number of rows = h
Form = rectangular
Type = complex single precision

2.3.43 Data Blocks Output From Module VDR

2.3.43.1 ØPHID (TABLE)

Description

Output complex eigenvectors requests (solution set, SØRT1, complex).

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	1014
	3	I	0
	4	I	Subcase number
	5	I	Mode number
	6-7	R	Complex eigenvalue
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in record 2 = 14.
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10 * point ID + device code (repeated for each point)
	2	I	Point type (repeated for each point)
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) (repeated for each point)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3) (repeated for each point)

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 9
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point
6. Components (words 3-14 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536

Word 2 = remainder from division above

Word 3-6 = zero.

2.3.43.2 ØUDVC1 (TABLE)

Description

Output displacement requests (solution set, SØRT1, complex)

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	1015 = displacement 1016 = velocity 1017 = acceleration
	3	I	0
	4	I	Subcase number
	5	R	Frequency
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code
	10	I	Number of words per entry in record 2 = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10 * point ID + divice code (repeated for each point)
	2	I	Point type (repeated for each point)
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) (repeated for each point)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3) (repeated for each point)

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginery; 3 = magnitude/phase
4. Approach code = 6

Notes cont'd.

5. Point type = 1 = grid point, 2 = scalar point, 3 = extra point, 4 = modal point

6. Components (words 3-14 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536.

Word 2 = remainder from division above.

Word 3-6 = zero.

2.3.43.3 ØUDV1 (TABLE)

Description

Output displacement requests (solution set, SØRT1, real).

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	15 = displacement, 16 = velocity, 17 = acceleration
	3	I	0
	4	I	Subcase number
	5	R	Time
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in record 2 = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10 * point ID + device code } repeated
	2	I	Point type } for each
	3-8	R	T1, T2, T3, R1, R2, R3 } point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = {0 = x y output only; 1 = print; 4 = punch; 5 = print and punch}

3. Format code = {1 = real; 2 = real/imaginary; 3 = magnitude/phase}

4. Approach code = 7

5. Point type = {1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point}

6. Components (words 3-8 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536.

Word 2 = remainder from division above.

Word 3-6 = zero.

2.3.43.4 ØPNL1 (TABLE)

Description

Output nonlinear load requests (solution set, SØRT1, real)

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	12
	3	I	0
	4	I	Subcase number
	5	R	Time
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code
	10	I	Number of words per entry in record 2 = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label

Record	Word	Type	Item+	
2	1	I	10 * point ID + device code	repeated for each point
	2	I	Point type	
	3-8	R	T1, T2, T3, R1, R2, R3	

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 7
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point
6. Components (words 3-8 in even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536.

Word 2 = remainder from division above.

Word 3-6 = zero.

2.3.43.5 ØPHIH (TABLE)

Description

Output complex eigenvector requests (solution set, SØRT1, complex).

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	1014
	3	I	0
	4	I	Subcase number
	5	I	Mode number
	6-7	R	Complex eigenvalue
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in record 2 = 14

Record	Word	Type	Item	
	11-50		Not defined	
	51-82	B	Title	
	83-114	B	Subtitle	
	115-146	B	Label	
2	1	I	10 * point ID + device code	repeated for each point
	2	I	Point type	
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)	
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)	

Notes

1. Records 1 and 2 are repeated for each vector to be output

2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch

3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase

4. Approach code = 9

5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

6. Components (words 3-14 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536.

Word 2 = remainder from division above.

Word 3-6 = zero.

2.3.43.6 ØUHVC1 (TABLE)

Description

Output displacement requests (solution set, SØRT1, complex).

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	1015 = displacement 1016 = velocity 1017 = acceleration
	3	I	0
	4	I	Subcase number
	5	R	Frequency
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code
	10	I	Number of words per entry in record 2 = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10 * point ID + device code (repeated for each point)
	2	I	Point type (repeated for each point)
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) (repeated for each point)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3) (repeated for each point)

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code =
   - 0 = x y output only
   - 1 = print
   - 4 = punch
   - 5 = print and punch

3. Format code =
   - 1 = real
   - 2 = real/imaginary
   - 3 = magnitude/phase

4. Approach code = 6

5. Point type =
   - 1 = grid point
   - 2 = scalar point
   - 3 = extra point
   - 4 = modal point

6. Components (words 3-14 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536.

Word 2 = remainder from division above.

Word 3-6 = zero.

2.3.43.7 ØUHV1 (TABLE)

Description

Output displacement requests (solution set, SØRT1, real).

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	15 = displacement 16 = velocity 17 = acceleration
	3	I	0
	4	I	Subcase number
	5	R	Time
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in record 2 = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10 * point ID + device code (repeated for each point)
	2	I	point type (repeated for each point)
	3-8	R	T1, T2, T3, R1, R2, R3 (repeated for each point)

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 7
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point
6. Components (words 3-8 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536.

Word 2 = remainder from division above.

Word 3-6 = zero.

2.3.44 Data Blocks Output From Module FRRD

2.3.44.1 UDVF (MATRIX)

Description

$[u_d^f]$ - Displacement vector matrix in a frequency response problem - d set.

Matrix Trailer

Number of columns	=	number of frequencies multiplied by the number of loads
Number of rows	=	d
Form	=	rectangular
Type	=	complex single precision

2.3.44.2 PSF (MATRIX)

Description

$[P_s^f]$ - Load vector for frequency response - s set.

Matrix Trailer

Number of columns	=	number of frequencies multiplied by the number of loads
Number of rows	=	s
Form	=	rectangular
Type	=	complex single precision

2.3.44.3 PDF (MATRIX)

Description

$\{P_d^f\}$ - Dynamic load matrix for frequency analysis - d set.

Matrix Trailer

Number of columns	=	number of frequencies multiplied by the number of loads
Number of rows	=	d
Form	=	rectangular
Type	=	complex single precision

2.3.44.4 PPF (MATRIX)

Description

$[P_p^f]$ - Dynamic loads for frequency response - p set.

Matrix Trailer

Number of columns	=	number of frequencies multiplied by the number of loads
Number of rows	=	p
Form	=	rectangular
Type	=	complex single precision

Note

The header record contains the list of frequencies.

2.3.44.5 UHVF (MATRIX)

Description

$[u_h^f]$ - Modal frequency response solution vectors - h set.

Matrix Trailer

Number of columns	=	number of frequencies multiplied by the number of loads
Number of rows	=	h
Form	=	rectangular
Type	=	complex single precision

2.3.45 Data Blocks Output From Module SDR3.

2.3.45.1 ØPP2 (TABLE)

Description

Output load vector requests (p set, SØRT2, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2002
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time — repeat for each time step
	2	I	Point type — repeat for each time step
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) — repeat for each time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 6
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.45.2 ØQP2 (TABLE).

Description

Output forces of single-point constraint (p set, SØRT2, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2003
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time — repeat for each time step
	2	I	Point type — repeat for each time step
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) — repeat for each time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 6
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.45.3 ØUPV2 (TABLE).

Description

Output displacement vector requests (p set, SØRT2, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{2001 = Displacement 2010 = Velocity 2011 = Acceleration}
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time } repeat
	2	I	Point type } for each
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = {0 = x y output only
1 = print
4 = punch
5 = print and punch}
3. Format code = {1 = real
2 = real/imaginary
3 = magnitude/phase}
4. Approach code = 6
5. Point type = {1 = grid point
2 = scalar point
3 = extra point
4 = modal point}

Table Trailer

Words 1-6 contain no significant values.

2.3.45.4 ØES2 (TABLE).

Description

Output element stress requests (SØRT2, real).

Table Format

Record	Word	Type	Item	
0			Header record	
1	1	I	Device code + 10*approach code	
	2	I	2005	
	3	I	Element type	
	4	I	Subcase number	
	5	I	10*element ID + device code	
	6	I	0	
	7	I	0	
	8	I	Dynamic load set ID	
	9	I	Format code = 1	
	10	I	Number of words per entry in next record = NWDS	
	11-50		Not defined	
	51-82	B	Title	
	83-114	B	Subtitle	
	115-146	B	Label	
2	1	R	Time	repeat for each time step
	2-NWDS	Mixed	Element stress data See section 2.3.51 for details	repeat for each time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 6

Table Trailer

Words 1-6 contain no significant values.

2.3.45.5 ØEF2 (TABLE).

Description

Output element force requests (SØRT2, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2004
	3	I	Element type
	4	I	Subcase number
	5	I	10*element ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time } repeat
	2-NWDS	Mixed	Element force data } for each
			See section 2.3.52 for details } time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 6

Table Trailer

Words 1-6 contain no significant values.

2.3.45.6 ØPNL2 (TABLE).

Description

Output nonlinear load requests (solution set, SØRT2, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2012
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time } repeat
	2	I	Point type } for each
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 6

5. Point type = { 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point }

6. Components (words 3-8 in even numbered records) which are not in the solution set are replaced by integer 1.

Table Trailer

Words 1-6 contain no significant values.

2.3.45.7 ØUDV2 (TABLE).

Description

Output displacement vector requests (solution set, SØRT2, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{2015 = Displacement 2016 = Velocity 2017 = Acceleration
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time } repeat
	2	I	Point type } for each
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = {0 = x y output only
1 = print
4 = punch
5 = print and punch
3. Format code = {1 = real
2 = real/imaginary
3 = magnitude/phase
4. Approach code = 6
5. Point type = {1 = grid point
2 = scalar point
3 = extra point
4 = modal point
6. Components (words 3-8 of even numbered records) which are not in the solution set are replaced by integer 1.

Table Trailer

Words 1-6 contain no significant values.

2.3.45.8 ØUHV2 (TABLE).

Description

Output displacement vector requests (solution set, SØRT2, real).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2015 = Displacement 2016 = Velocity 2017 = Acceleration
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time — repeat for each time step
	2	I	Point type — repeat for each time step
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) — repeat for each time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 6
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point
6. Components (words 3-8 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Words 1-6 contain no significant values.

2.3.45.9 ØPPC2 (TABLE).

Description

Output load vector requests (p set, SØRT2, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3002
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency (repeat for each frequency)
	2	I	Point type (repeat for each frequency)
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) (repeat for each frequency)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3) (repeat for each frequency)

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 5
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.45.10 ØQPC2 (TABLE).

Description

Output forces of single-point constraint requests (p set, SØRT2, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3003
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency (repeat for each frequency)
	2	I	Point type (repeat for each frequency)
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) (repeat for each frequency)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3) (repeat for each frequency)

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 5
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point

Table Trailer

Words 1-6 contain no significant values.

2.3.45.11 ØUPVC2 (TABLE).

Description

Output displacement vector requests (p set, SØRT2, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{3001 = Displacement 3010 = Velocity 3011 = Acceleration
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency — repeat for each frequency
	2	I	Point type — repeat for each frequency
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) — repeat for each frequency
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3) — repeat for each frequency

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = {0 = x y output only; 1 = print; 4 = punch; 5 = print and punch}
3. Format code = {1 = real; 2 = real/imaginary; 3 = magnitude/phase}
4. Approach code = 5
5. Point type = {1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point}

Table Trailer

Words 1-6 contain no significant values.

2.3.45.12 ØESC2 (TABLE).

Description

Output element stress requests (SØRT2, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3005
	3	I	Element type
	4	I	Subcase number
	5	I	10*Element ID + device code
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency } repeat
	2-NWDS	Mixed	Element stress data } for each
			See 2.3.51 for details } frequency

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 5

Table Trailer

Words 1-6 contain no significant values.

2.3.45.13 ØEFC2 (TABLE).

Description

Output element force requests (SØRT2, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3004
	3	I	Element type
	4	I	Subcase number
	5	I	10*element ID + device code
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency } repeat
	2-NWDS	Mixed	Element force data } for each
			See 2.3.52 for details } frequency

Notes

Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch }

3. Format code = { 1 = real; 2 = real/imaginary; 3 = magnitude/phase }

4. Approach code = 5

Table Trailer

Words 1-6 contain no significant values.

2.3.45.14 ØUDVC2 (TABLE).

Description

Output displacement vector requests (solution set, SØRT2, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3015 = Displacement 3016 = Velocity 3017 = Acceleration
	3	I	0
	4	I	Subcase Number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency — repeat for each frequency
	2	I	Point type — repeat for each frequency
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) — repeat for each frequency
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3) — repeat for each frequency

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 5
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point
6. Components (words 3-14 of even numbered records) which are not in the solution set are replaced by integer 1.

Table Trailer

Words 1-6 contain no significant values.

2.3.45.15 ØUHVC2 (TABLE).

Description

Output displacement vector requests (solution set, SØRT2, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3015 = Displacement 3016 = Velocity 3017 = Acceleration
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency (repeat for each frequency)
	2	I	Point type (repeat for each frequency)
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) (repeat for each frequency)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3) (repeat for each frequency)

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = 0 = x y output only; 1 = print; 4 = punch; 5 = print and punch
3. Format code = 1 = real; 2 = real/imaginary; 3 = magnitude/phase
4. Approach code = 5
5. Point type = 1 = grid point; 2 = scalar point; 3 = extra point; 4 = modal point
6. Components (words 3-14 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Words 1-6 contain no significant values.

2.3.46 Data Blocks Output From Module XYTRAN.

2.3.46.1 XYPLTT (TABLE).

Description

Output plot request data in form for direct plotting of SØRT2 Transient Response output.

Table Format

Record	Word	Type	Item
0			Header record
1	1	I-R	Subcase ID or if a Random Response problem, the Mean Response
	2	I	Frame number
	3	I	Curve number
	4	I	Point ID or element ID
	5	I	Component number
	6	I	Vector number (1, 2, ... 11)
	7	I	1--Graph uses top half of frame 0--Graph uses full frame -1--Graph uses lower half of frame
	8	I	0--Axis, tics, labels, values, etc. have been drawn and this curve is to be scaled and plotted identically as last except for curve symbols. 1--Axis, tics, labels, scaling, etc. are to be performed or computed and if word 7 of this record = 0 or 1, a skip to new frame is to be made.
	9	I	Number of blank frames between frames (frame-skip)
	10		Not used
	11	R	XMIN }
	12	R	XMAN } Define exact corners of this
	13	R	YMIN } upper, lower, or full frame graph
	14	R	YMAX }
	15	R	Actual value of first tic }
	16	R	Actual increment to successive tics }
	17	I	Integer value to be printed on first tic }
	18	I	Maximum number of digits in any print-value } x-direction
	19	I	+ or - power for print values } tics
	20	I	Total number of tics to print this edge }
	21	I	Value print skip 0,1,2,3--- }
	22	I	Delta integer print value to successive tics }
	23	R	}
	24	R	}
	25	I	}
	26	I	} Same as 15 through 22
	27	I	} But for y-direction tics
	28	I	}
	29	I	}
	30	I	}

Record	Word	Type	Item
	31	I	Top edge tics — each of 31-34 may be less than 0--tics without values; equal to 0--no tics here; greater 0--tics with values
	32	I	Bottom edge tics — each of 31-34 may be less than 0--tics without values; equal to 0--no tics here; greater 0--tics with values
	33	I	Left edge tics — each of 31-34 may be less than 0--tics without values; equal to 0--no tics here; greater 0--tics with values
	34	I	Right edge tics — each of 31-34 may be less than 0--tics without values; equal to 0--no tics here; greater 0--tics with values
	35	I	0--x-direction is linear Greater than 0--number of cycles and x-direction is logarithmic
	36	I	0--y-direction is linear Greater than 0--number of cycles and y-direction is logarithmic
	37	I	0--no x-axis 1--draw x-axis
	38	R	x-axis y-intercept
	39	I	0--no y-axis 1--draw y-axis
	40	R	y-axis x-intercept
	41	I	Less than 0 -- plot symbol for each curve point. Select symbol corresponding to curve number in word 3 of this record. Equal to 0 -- connect points by lines where points are continuous i.e., (no integer 1 pairs). Greater than 0 -- do both of above.
	42 . . . 50		Not used
	51	B	Title (32 words)
	.	B	Subtitle (32 words)
	.	B	Label (32 words)
	.	B	Curve Title (32 words)
	.	B	x-axis Title (32 words)
	242	B	y-axis Title (32 words)
	243 . . . 282		Not used
	283	I	Pensize
	284	I	Plotter 1 = SC4020, 2 = EAI3500
	285	R	Inches paper x-direction
	286	R	Inches paper y-direction
	287	I	Camera for SC4020 less than 0 = 35mm, 0 = F80, Greater 0 = Both
	288	I	Print flag — 0 = no, 1 = yes
	289	I	Plot flag — 0 = no, 1 = yes
	290	I	Punch flag — 0 = no, 1 = yes
	291	R	x-min of all data
	292	R	x-max of all data
	293	R	y-min within x-limits of graph
	294	R	x-value at this y-min
	295	R	y-max within x-limits of graph
	296	R	x-value at this y-max
	297	R	y-min for all data
	298	R	x-value at this y-min
	299	R	y-max for all data
	300	R	x-value at this y-max

Record	Word	Type	Item
2	1	I	1 } Always is present
	2	I	1 }
	3	R	x-value } coordinate pair
	4	R	y-value } repeats for all pairs plotted

Notes

1. Records 1 and 2 repeat for each curve plotted.

2. Even numbered records will contain integer 1 pairs to indicate where curve has moved outside of graph limits.

Table Trailer

Words 1-6 contain no significant values.

2.3.46.2 XYPLTFA (TABLE).

Description

Identical to XYPLTT, for Frequency Response plots (solution set).

2.3.46.3 XYPLTF (TABLE).

Description

Identical to XYPLTT, for Frequency Response plots.

2.3.46.4 XYPLTR (TABLE).

Description

Identical to XYPLTT, for Random Response plots.

2.3.46.5 XYPLTTA (TABLE).

Description

Identical to XYPLTT, for Transient Response plots (solution set).

2.3.47 Data Blocks Output From Module RANDØM

2.3.47.1 PSDF (TABLE)

Description

Power Spectral Density Table.

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	50
	2	I	Code for data type DISP = 2001 VELØ = 2010 ACCE = 2011 LØAD = 2002 SPCF = 2003 ELFØ = 2004 STRE = 2005
	3	I	4001
	4	I	0
	5	I	Point or element ID times 10
	6	I	Component ID + 2
	7	I	0
	8	R	Mean response
	9	I	0
	10	I	2
	11-50	I	0
	51-146	B	Title, subtitle, label
2	1	R	Frequency
	2	R	Power spectral density

Notes

1. Words 1 and 2 of record 2 are repeated for each frequency.
2. Records 1 and 2 are repeated for each power spectral density request.

Table Trailer

Words 1-5 = zero.

Word 6 = number of requests.

2.3.47.2 AUTØ (TABLE).

Description

Autocorrelation function table.

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	50
	2	I	Code for data type DISP = 2001 VELØ = 2010 ACCE = 2011 LØAD = 2002 SPCF = 2003 ELFØ = 2004 STRE = 2005
	3	I	4002
	4	I	0
	5	I	Point or element ID times 10
	6	I	Component ID + 2
	7	I	0
	8	R	Mean response
	9	I	0
	10	I	2
	11-50	I	0
	51-146	B	Title, subtitle, label
2	1	R	TAU
	2	R	Auto correlation function

Notes

1. Words 1 and 2 of record 2 are repeated for each TAU.
2. Records 1 and 2 are repeated for each autocorrelation request.

Table Trailer

Words 1-5 = zero.

Word 6 = number of requests.

2.3.48 Data Blocks Output From Module TRD.

2.3.48.1 UDVT (MATRIX)

Description

$[u_d^t]$ - Displacement, velocity, and acceleration vector matrix in a transient analysis problem - d set.

Matrix Trailer

Number of columns	=	three times the number of output time steps
Number of rows	=	d
Form	=	rectangular
Type	=	real single precision

2.3.48.2 PDT (MATRIX)

Description

$[p_d^t]$ - Linear dynamic load matrix for transient analysis - d set.

Matrix Trailer

Number of columns	=	number of output times
Number of rows	=	d
Form	=	rectangular
Type	=	real single precision

2.3.48.3 PST (MATRIX)

Description

$[p_s^t]$ - Linear load vector for transient analysis - s set.

Matrix Trailer

Number of columns	=	number of output times
Number of rows	=	s
Form	=	rectanqular
Type	=	real single precision

2.3.48.4 PPT (MATRIX)

Description

$[p_p^t]$ - Linear dynamic loads for transient analysis - p set.

Matrix Trailer

Number of columns = number of output times
Number of rows = p
Form = rectangular
Type = real single precision

*Note: The header record contains the list of output times.

2.3.48.5 PNLD (MATRIX)

Description

$[p_d^{n\ell}]$ - Nonlinear loads in transient problem - d set.

Matrix Trailer

Number of columns = number of output times
Number of rows = d
Form = rectangular
Type = real single precision

2.3.48.6 UHVT (MATRIX)

Description

$[u_h^t]$ - Modal transient solution vectors - h set.

Matrix Trailer

Number of columns = three times the number of output times
Number of rows = h
Form = rectangular
Type = real single precision

2.3.48.7 PNLH (MATRIX)

Description

$[p_h^{n\ell}]$ - Nonlinear loads in modal transient problem - h set.

Matrix Trailer

Number of columns = number of output times
Number of rows = h
Form = rectangular
Type = real single precision

2.3.49 Data Blocks Output From Module GKAM

2.3.49.1 MHH (MATRIX)

Description

$[m_{hh}]$ - Modal mass matrix - h set.

Matrix Trailer

Number of columns	=	h
Number of rows	=	h
Form	=	symmetric
Type	=	real double precision

2.3.49.2 BHH (MATRIX)

Description

$[b_{hh}]$ - Modal damping matrix - h set.

Matrix Trailer

Number of columns	=	h
Number of rows	=	h
Form	=	symmetric
Type	=	real double precision

2.3.49.3 KHH (MATRIX)

Description

$[k_{hh}]$ - Modal stiffness matrix - h set.

Matrix Trailer

Number of columns	=	h
Number of rows	=	h
Form	=	symmetric
Type	=	real double precision

2.3.49.4 PHIDH (MATRIX)

Description

$[\phi_{dh}]$ - Transformation matrix from d set to modal coordinates.

Matrix Trailer

Number of columns	=	h
Number of rows	=	d
Form	=	rectangular
Type	=	real single precision

2.3.50 Data Blocks Output From Module DDR1

2.3.50.1 CPHID (MATRIX)

Description

$[\Phi_d]$ - Complex eigenvector matrix transformed from modal to physical coordinates.

Matrix Trailer

Number of columns	=	number of eigenvectors
Number of rows	=	d
Form	=	rectangular
Type	=	complex single precision

2.3.50.2 UDV1F (MATRIX)

Description

$[u_d^f]$ - Displacement vectors matrix in a frequency response problem - d set.

Matrix Trailer

Number of columns	=	number of frequencies times number of loads
Number of rows	=	d
Form	=	rectangular
Type	=	complex single precision

2.3.50.3 UDV2F (MATRIX)

Description

$[u_d^t]$ - Displacement vectors matrix in a transient response problem - d set.

Matrix Trailer

Number of columns	=	number of output times multiplied by 3
Number of rows	=	d
Form	=	rectangular
Type	=	real single precision

2.3.51 Element Stress Output Data Description.

Note particular data block description (e.g., ØES1, ØESB1) for contents of word 1 for each element.

Element		Real Element Stresses		Complex Element Stresses		
Type	Name	Word or Component	Item	Word or Component	Item	Real Imag.
1	CRØD	2	Axial	2	Axial	R
		3	Axial Safety Margin *	3	Axial	I
		4	Torsional	4	Torsional	R
		5	Torsional Safety Margin*	5	Torsional	I
2	CBEAM	2	SA1		Undefined	
		3	SA2			
		4	SA3			
		5	Axial			
		6	SA-maximum			
		7	SA-minimum			
		8	Safety Margin in Tension*			
		9	SB1			
		10	SB2			
		11	SB3			
		12	SB-maximum			
		13	SB-minimum			
		14	Safety Margin in Comp*			
3	CTUBE		Same as CRØD		Same as CRØD	
4	CSHEAR	2	Maximum Shear	2	Maximum Shear	R
		3	Average Shear	3	Maximum Shear	I
		4	Safety Margin*	4	Average Shear	R
				5	Average Shear	I
5	CTWIST	2	Maximum	2	Maximum	R
		3	Average	3	Maximum	I
		4	Safety Margin*	4	Average	R
				5	Average	I
6	CTRIA1	2	Z1 = Fibre Distance 1	2	Z1 = Fibre Distance 1	
		3	Normal-x at Z1	3	Normal-x at 1	R
		4	Normal-y at Z1	4	Normal-x at 1	I
		5	Shear-xy at Z1	5	Normal-y at 1	R
		6	0-Shear Angle at Z1	6	Normal-y at 1	I
		7	Major-Principal at Z1	7	Shear-xy at 1	R
		8	Minor-Principal at Z1	8	Shear-xy at 1	I
		9	Max-Shear at Z1	9	Z2 = Fibre Distance 2	
		10	Z2 = Fibre Distance 2	10	Normal-x at 2	R
		11	Normal-x at Z2	11	Normal-x at 2	I
		12	Normal-y at Z2	12	Normal-y at 2	R
		13	Shear-xy at Z2	13	Normal-y at 2	I
		14	0-Shear Angle at Z2	14	Shear-xy at 2	R
		15	Major-Principal at Z2	15	Shear-xy at 2	I
		16	Minor-Principal at Z2			
		17	Maximum-Shear at Z2			

Element		Real Element Stresses		Complex Element Stresses		
Type	Name	Word or Component	Item	Word or Component	Item	Real Imag.
7	CTRBSC		Note CTRIA1		Note CTRIA1	
8	CTRPLT		Note CTRIA1		Note CTRIA1	
9	CTRMEM	2	Normal-x	2	Normal-x	R
		3	Normal-y	3	Normal-x	I
		4	Shear-xy	4	Normal-y	R
		5	0-Shear Angle	5	Normal-y	I
		6	Major-Principal	6	Shear-xy	R
		7	Minor-Principal	7	Shear-xy	I
		8	Maximum Shear			
10	CØNRØD		Note CRØD		Note CRØD	
11	CELAS1	2	Stress	2	Stress	R
				3	Stress	I
12	CELAS2	2	Stress	2	Stress	R
				3	Stress	I
13	CELAS3	2	Stress	2	Stress	R
				3	Stress	I
14	CELAS4		Undefined		Undefined	
15	CQDPLT		Note CTRIA1		Note CTRIA1	
16	CQDMEM		Note CTRMEM		Note CTRMEM	
17	CTRIA2		Note CTRIA1		Note CTRIA1	
18	CQUAD2		Note CTRIA1		Note CTRIA1	
19	CQUAD1		Note CTRIA1		Note CTRIA1	
20	CDAMP1		Undefined		Undefined	
21	CDAMP2		Undefined		Undefined	
22	CDAMP3		Undefined		Undefined	
23	CDAMP4		Undefined		Undefined	
24	CVISC		Undefined		Undefined	
25	CMASS1		Undefined		Undefined	
26	CMASS2		Undefined		Undefined	
27	CMASS3		Undefined		Undefined	
28	CMASS4		Undefined		Undefined	
29	CØNM1		Undefined		Undefined	
30	CØNM2		Undefined		Undefined	
31	CPLØTEL		Undefined		Undefined	
32	CREACT		Undefined		Undefined	

Element		Real Element Stresses		Complex Element Stresses		
Type	Name	Word or Component	Item	Word or Component	Item	Real Imag.
33	CQUAD3		Undefined		Undefined	
34	CBAR	2	SA1	2	SA1	R
		3	SA2	3	SA2	R
		4	SA3	4	SA3	R
		5	SA4	5	SA4	R
		6	Axial	6	Axial	R
		7	SA-maximum	7	SA1	I
		8	SA-minimum	8	SA2	I
		9	Safety Margin in Tension*	9	SA3	I
		10	SB1	10	SA4	I
		11	SB2	11	Axial	I
		12	SB3	12	SB1	R
		13	SB4	13	SB2	R
		14	SB-maximum	14	SB3	R
		15	SB-minimum	15	SB4	R
		16	Safety Margin in Comp*	16	SB1	I
				17	SB2	I
				18	SB3	I
				19	SB4	I
35	CCØNEAX	2	Harmonic or point angle		Undefined	
		3	Z1 = Fibre Distance 1			
		4	Normal-u at 1			
		5	Normal-v at 1			
		6	Shear-uv at 1			
		7	0-Shear Angle at 1			
		8	Major-Principal at 1			
		9	Minor-Principal at 1			
		10	Maximum Shear at 1			
		11	Z2 = Fibre Distance 2			
		12	Normal-u at 2			
		13	Normal-v at 2			
		14	Shear-uv at 2			
		15	0-Shear Angle at 2			
		16	Major-Principal at 2			
		17	Minor-Principal at 2			
		18	Maximum-Shear at 2			
36	CTRIARG	2	Radial (x)		Undefined	
		3	Circum (Theta)			
		4	Axial (z)			
		5	Shear (zx)			
37	CTRAPRG	2	Radial (x) at 1		Undefined	
		3	Circum (Theta) at 1			
		4	Axial (z) at 1			
		5	Shear (zx) at 1			
		6	Radial (x) at 2			
		7	Circum (Theta) at 2			
		8	Axial (z) at 2			
		9	Shear (zx) at 2			
		10	Radial (x) at 3			
		11	Circum (Theta) at 3			
		12	Axial (z) at 3			
		13	Shear (zx) at 3			
		14	Radial (x) at 4			
		15	Circum (Theta) at 4			
		16	Axial (z) at 4			
		17	Shear (zx) at 4			
		18	Radial (x) at 5			

Element Type	Element Name	Real Element Stresses Word or Component	Item		Complex Element Stresses Word or Component	Item	Real Imag.
37 cont'd.		19	Circum(Theta)	at 5			
		20	Axial (z)	at 5			
		21	Shear (zx)	at 5			
38	CTØRDRG	2	Mem.-Tagen.	at 1		Undefined	
		3	Mem.-Circum.	at 1			
		4	Flex.-Tangen.	at 1			
		5	Flex.-Circum.	at 1			
		6	Shear-Force	at 1			
		7	Mem.-Tangen.	at 2			
		8	Mem.-Circum.	at 2			
		9	Flex.-Tangen.	at 2			
		10	Flex.-Circum.	at 2			
		11	Shear-Force	at 2			
		12	Mem.-Tangen.	at 3			
		13	Mem.-Circum.	at 3			
		14	Flex.-Tangen.	at 3			
		15	Flex.-Circum.	at 3			
		16	Shear-Force	at 3			

* If not equal to integer 1.

Note

If output is magnitude/phase the magnitude replaces the real part and the phase replaces the imaginary part.

2.3.52 Element Force Output Data Description.

Note particular data block description (e.g., ØEF1, ØEFB1) for contents word 1 for each element.

Element Type	Element Name	Real Element Forces Word or Component	Item		Complex Element Forces Word or Component	Item	Real Imag.
1	CRØD	2	Axial Force		2	Axial Force	R
		3	Torque		3	Axial Force	I
					4	Torque	R
					5	Torque	I
2	CBEAM	2	Bend-Mom	A1		Undefined	
		3	Bend-Mom	A2			
		4	Bend-Mom	B1			
		5	Bend-Mom	B2			
		6	Shear-1				
		7	Shear-2				
		8	Axial Force				
		9	Torque				
3	CTUBE		Same as CRØD			Same as CRØD	
4	CSHEAR	2	Force Pts 1,3		2	Force Pts 1,3	R
		3	Force Pts 2,4		3	Force Pts 1,3	I
					4	Force Pts 2,4	R
					5	Force Pts 2,4	I
5	CTWIST	2	Moment Pts 1,3		2	Moment Pts 1,3	R
		3	Moment Pts 2,4		3	Moment Pts 1,3	I
					4	Moment Pts 2,4	R
					5	Moment Pts 2,4	I
6	CTRIA1	2	Bend-Mom-x		2	Bend-Mom-x	R
		3	Bend-Mom-y		3	Bend-Mom-y	R
		4	Twist-Moment		4	Twist-Moment	R
		5	Shear-x		5	Shear-x	R
		6	Shear-y		6	Shear-y	R
					7	Bend-Mom-x	I
					8	Bend-Mom-y	I
					9	Twist-Moment	I
					10	Shear-x	I
					11	Shear-y	I
7	CTRBSC		Same as CTRIA1			Same as CTRIA1	
8	CTRPLT		Same as CTRIA1			Same as CTRIA1	
9	CTRMEM		Undefined			Undefined	
10	CØNRØD		Same as CRØD			Same as CRØD	
11	CELAS1	2	Force		2	Force	R
					3	Force	I
12	CELAS2	2	Force		2	Force	R
					3	Force	I
13	CELAS3	2	Force		2	Force	R
					3	Force	I

Element		Real Element Forces			Complex Element Forces			
Type	Name	Word or Component	Item		Word or Component	Item		Real Imag.
14	CELAS4	2	Force		2 3	Force Force		R I
15	CQDPLT		Note CTRIA1			Note CTRIA1		
16	CQDMEM		Undefined			Undefined		
17	CTRIA2		Note CTRIA1			Note CTRIA1		
18	CQUAD2		Note CTRIA1			Note CTRIA1		
19	CQUAD1		Note CTRIA1			Note CTRIA1		
20	CDAMP1		Undefined			Undefined		
21	CDAMP2		Undefined			Undefined		
22	CDAMP3		Undefined			Undefined		
23	CDAMP4		Undefined			Undefined		
24	CVISC		Undefined			Undefined		
25	CMASS1		Undefined			Undefined		
26	CMASS2		Undefined			Undefined		
27	CMASS3		Undefined			Undefined		
28	CMASS4		Undefined			Undefined		
29	CØNM1		Undefined			Undefined		
30	CØNM2		Undefined			Undefined		
31	CPLØTEL		Undefined			Undefined		
32	CREACT		Undefined			Undefined		
33	CQUAD3		Undefined			Undefined		
34	CBAR	2 3 4 5 6 7 8 9	Bend-Mom Bend-Mom Bend-Mom Bend-Mom Shear-1 Shear-2 Axial Force Torque	A1 A2 B1 B2	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	Bend-Mom Bend-Mom Bend-Mom Bend-Mom Shear-1 Shear-2 Axial Force Torque Bend-Mom Bend-Mom Bend-Mom Bend-Mom Shear-1 Shear-2 Axial Force Torque	A1 A2 B1 B2 A1 A2 B1 B2 	R R R R R R R R I I I I I I I I

Element		Real Element Forces		Complex Element Forces		
Type	Name	Word or Component	Item	Word or Component	Item	Real Imag.
35	CCØNEAX	2	Harmonic or point angle		Undefined	
		3	Bend-Mom u			
		4	Bend-Mom v			
		5	Twist-Moment			
		6	Shear u			
		7	Shear v			
36	CTRIARG	2	Radial (x) at 1		Undefined	
		3	Circum (Theta) at 1			
		4	Axial (z) at 1			
		5	Radial (x) at 2			
		6	Circum (Theta) at 2			
		7	Axial (z) at 2			
		8	Radial (x) at 3			
		9	Circum (Theta) at 3			
		10	Axial (z) at 3			
37	CTRAPRG	2	Radial (x) at 1		Undefined	
		3	Circum (Theta) at 1			
		4	Axial (z) at 1			
		5	Radial (x) at 2			
		6	Circum (Theta) at 2			
		7	Axial (z) at 2			
		8	Radial (x) at 3			
		9	Circum (Theta) at 3			
		10	Axial (z) at 3			
		11	Radial (x) at 4			
		12	Circum (Theta) at 4			
		13	Axial (z) at 4			
38	CTØRDRG	2	Radial (x) at 1		Undefined	
		3	Circum (Theta) at 1			
		4	Axial (z) at 1			
		5	Moment (zx) at 1			
		6	Direct Strain at 1			
		7	Curvature at 1			
		8	Radial (x) at 2			
		9	Circum (Theta) at 2			
		10	Axial (z) at 2			
		11	Moment (zx) at 2			
		12	Direct Strain at 2			
		13	Curvature at 2			

2.3.53 Data Blocks Output From Module DDR2

2.3.53.1 UEVF (MATRIX)

Description

$[u_e^f]$ - Displacements at the extra points for a frequency response problem.

Matrix Trailer

Number of columns	=	number of frequencies times number of loads
Number of rows	=	e
Form	=	rectangular
Type	=	single precision

2.3.53.2 PAF (MATRIX)

Description

$[P_a^f]$ - Equivalent load vector for mode acceleration computations in a frequency response problem - a set.

Matrix Trailer

Number of columns	=	number of frequencies times number of loads
Number of rows	=	d
Form	=	rectangular
Type	=	single precision

2.3.53.3 UDV2F (MATRIX)

Description

$[u_d^{fa}]$ - Mode accelerated displacement vectors for a frequency response problem.

Matrix Trailer

Number of columns	=	number of frequencies times number of loads
Number of rows	=	d
Form	=	rectangular
Type	=	complex single precision

2.3.53.4 UEVT (MATRIX)

Description

$[u_e^t]$ - Displacement at the extra points for a transient analysis problem.

Matrix Trailer

Number of columns	=	number of output times multiplied by 3
Number of rows	=	e
Form	=	rectangular
Type	=	real single precision

2.3.53.5 PAT (MATRIX)

Description

$[P_a^t]$ - Equivalent load vector for mode acceleration in a transient analysis problem.

Matrix Trailer

Number of columns	-	number of output times multiplied by 3
Number of rows	-	d
Form	-	rectangular
Type	-	real single precision

2.3.53.6 UDV2T (MATRIX)

Description

$[u_d^{ta}]$ - Mode accelerated displacement vectors for a transient analysis problem.

Matrix Trailer

Number of columns	=	number of output times multiplied by 3
Number of rows	=	d
Form	=	rectangular
Type	=	real single precision

2.4 EXECUTIVE TABLE DESCRIPTIONS

The following is an alphabetical index of Executive table descriptions.

Section Number	Executive Table Name	Where Stored	Page Number
2.4.1.5	CEITBL	/XCEITB/	2.4-9
2.4.1.4	DPL	/XDPL/	2.4-7
2.4.1.2	FIAT	/XFIAT/	2.4-3
2.4.1.3	FIST	/XFIST/ and /XPFIST/	2.4-5
2.4.2.7	LNKSPC	/XLKSPC/	2.4-29
2.4.2.2	MPL	/XGPI2/	2.4-21
2.4.2.1	ØSCAR	Data Pool File	2.4-15
2.4.2.4	PVT	/XPVT/	2.4-26
2.4.1.8	SYSTEM	/SYSTEM/	2.4-13
2.4.1.7	TAPID	/STAPID/	2.4-12
2.4.1.6	VPS	/XVPS/	2.4-10
2.4.2.6	XALTER	Problem Tape	2.4-28
2.4.2.5	XCSA	Problem Tape	2.4-27
2.4.1.1	XFIAT	/XXFIAT/	2.4-2
2.4.1.9	XLINK	/XLINK/	2.4-14
2.4.2.3	XPTDIC	Problem Tape	2.4-24

2.4.1 Executive Tables Which are Permanently Core Resident

2.4.1.1 XFIAT (Permanent File Allocation Table)

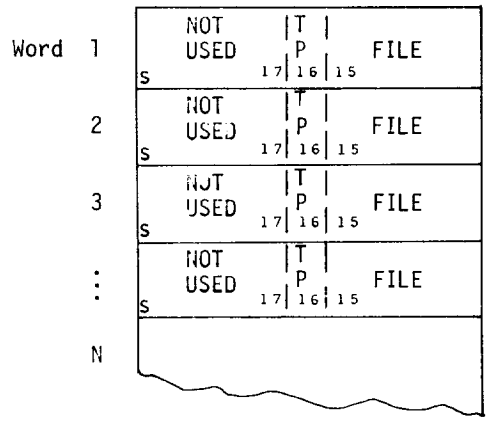
Description

A NASTRAN resident memory table containing the physical file identification for the permanent files (PØØL, ØPTP, etc.).

Created in Module

The physical file identifications are output by GNFIAT (generate FIAT).

Table Format



Word	Item	Description
1-N	TP	Tape Flag (1 bit) - set if physical file (FILE) is a magnetic tape.
	FILE	File ID (15 bits) - unique integer identification for a physical file.

Notes

1. The number of entries (N) is dictated by the integer value in PFIST (see FIST Executive Table Description - 2.4.1.3)
2. The XFIAT table is located in the named common block /XXFIAT/.

2.4.1.2 FIAT (File Allocation Table).

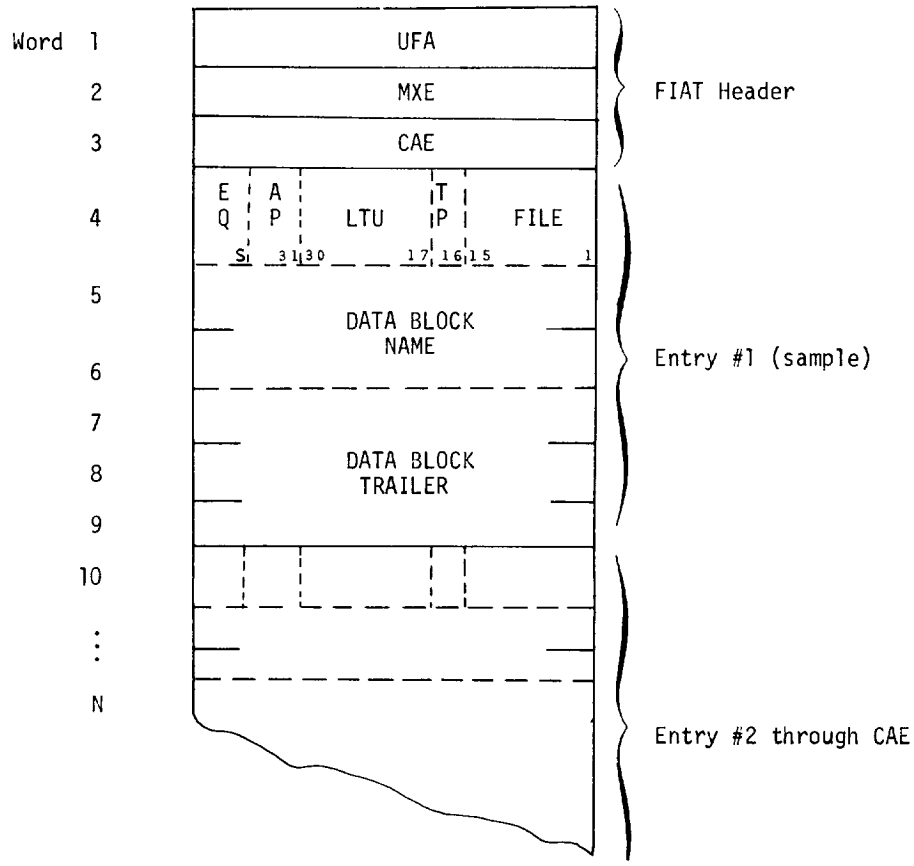
Description

A NASTRAN resident memory table containing the data block name vs. physical file ID. for a segment of DMAP modules.

Created in Module

The physical files available for the system/computer configuration are output by GNFIAT (generate FIAT). The data block names and other data block information are output by XSFA, Executive Segment File Allocator.

Table Format



Word	Item	Description
1	UFA	Unique files available - this integer indicates the number of unique file entries in the FIAT.
2	MXE	Maximum entries - this integer shows the total entry size of the dimensioned FIAT table; the amount of memory reserved (N) = 6 x MXE + 3.
3	CAE	Current active entries - this integer designates the portion of FIAT currently containing valid data; UFA ≤ CAE ≤ MXE.

Words 4 through 9 describe a sample 6-word entry:

4	EQ	Equivalence flag (1 bit) - 0 bit, file not equivalenced. 1 bit, file equivalenced.
	AP	Append flag (1 bit) - set if append is specified for data block in DMAP sequence by a FILE DMAP instruction.
	LTU	Last time used (14 bit integer) - record number of ØSCAR entry for last use of data block.
	TP	Tape flag (1 bit) - set if physical file (FILE) is a magnetic tape.
	FILE	File ID (15 bits) - unique identification for a physical file.
5,6	NAME	Data block name - 8 characters (4 characters/word).
7,8,9	TRAILER	Data block trailer - storage for 6-16 bit data block trailer words.

Words 10 through N contain repeated 6-word entries.

Trailer Information

Trailer information for each data block is stored in and received from the FIAT by WRTTRL (write trailer) and RDTRL (read trailer).

Note

The FIAT table is located in the named common block /XFIAT/.

2.4.1.3 FIST (File Status Table)

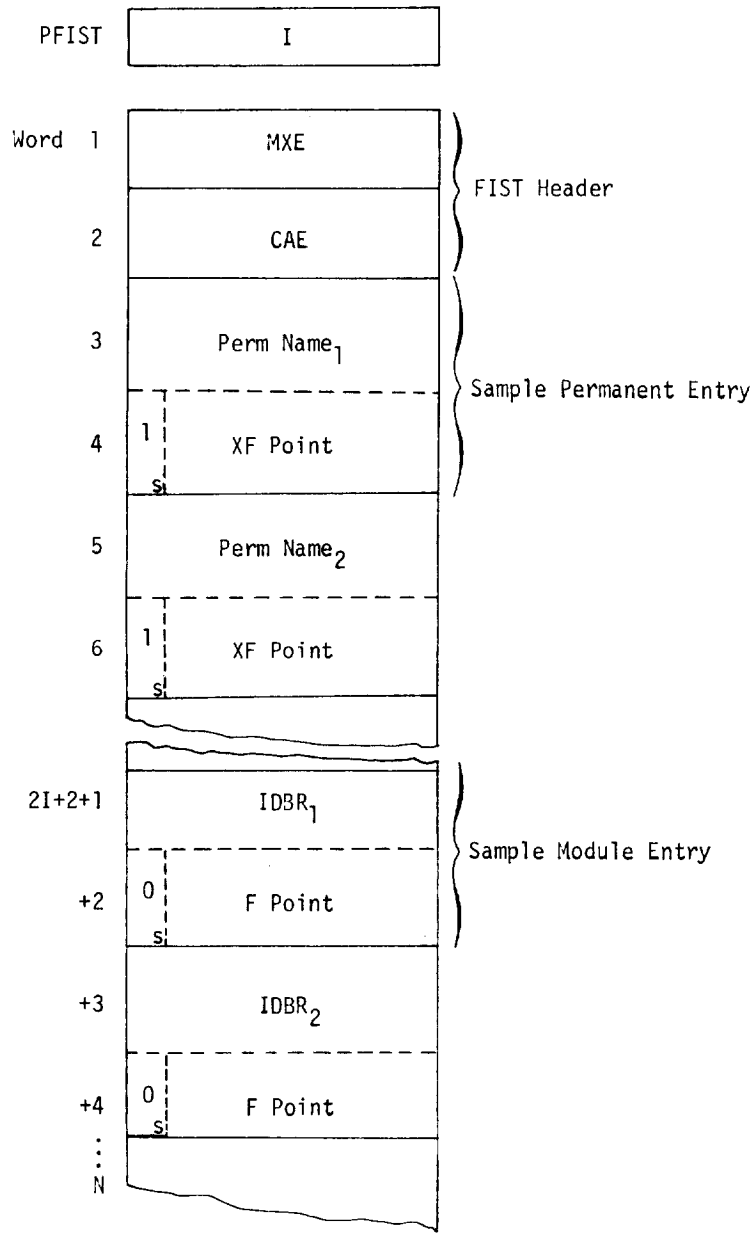
Description

A NASTRAN resident memory table containing the internal data block reference (IDBR) numbers vs. FIAT table pointers for a particular module; also, the permanent file reference names vs. XFIAT table pointers.

Created in Module

The module entries are generated prior to each module execution by subroutines GNFIST (Generate FIST). The permanent entries are initialized at system assembly.

Table Format



Word	Item	Description
PFIST	I	Integer number of permanent FIST entries.
1	MXE	Maximum entries - this integer shows the total entry size of the dimensioned FIST table; the amount of memory reserved (N) = 2 * MXE + 2.
2	CAE	Current active entries - this integer designates the portion of FIST currently containing valid data; I ≤ CAE ≤ MXE.

Words 3 and 4 describe a sample 2-word permanent entry:

3	Perm Name	A permanent file reference name - 4 characters BCD (e.g., PØØL, ØPTP, PLT1, etc.).
4	XF Point	Points to the XFIAT position containing the file ID for this permanent file.

Words 2I+2+1 and +2 describe a sample 2-word module entry.

+1	IDBR	An internal data block reference number (GINØ file number) - integer (e.g., 104, 206, 301, etc.).
+2	F Point	Points to the FIAT position containing the file ID for this module entry.

Notes

1. XFIAT pointer values contain an S bit equal to 1, while FIAT pointer values contain an S bit equal to 0.
2. Permanent entries remain static throughout a run, while module entries are changed by GNFIST prior to each module call.
3. FIAT and XFIAT position pointers are indexes into the respective tables considering the first word of the table as position 0.
4. The FIST table is located in the named common block /XFIST/.
   The PFIST entry is located in the named common block /XPFIST/.

2.4.1.4 DPL (Data Pool Dictionary)

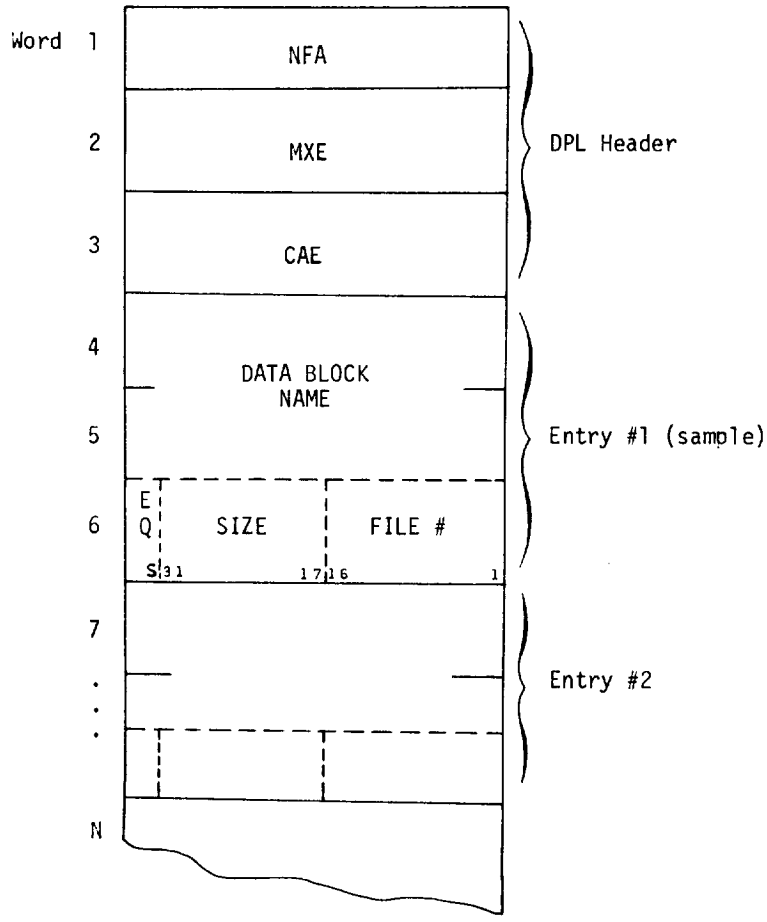
Description

A NASTRAN resident memory table describing the current contents and status of the Data Pool.

Created in Module

Data Pool, and therefore Dictionary entries, are created by pooling from SFA (Segment File Allocator), housekeeping operations by DPH (Data Pool Housekeeper) and restart initialization by GPI (General Problem Initialization), and IFP (Input File Processor) when writing DMI and DTI information (see section 2.3.2).

Table Format



Word	Item	Description
1	NFA	Next file available - the next Data Pool File number (integer) available for output.
2	MXE	Maximum entries - this integer shows the total entry size of the dimensioned DPL table; the amount of memory reserved (N) = 3 * MXE + 3.
3	CAE	Current active entries - this integer designates the portion of the DPL currently containing valid data; 0 ≤ CAE ≤ MXE.

Words 4 through 6 describe a sample 3-word entry.

4,5	NAME	Data block name - 8 characters (4 characters/word).
6	EQ	Equivalence flag (1 bit) - 0 bit, file not equivalenced. 1 bit, file equivalenced.
	SIZE	Size of the pooled data block - number of words/10.
	FILE#	The file number (integer) showing the relative position of the data block file of the pool.

Note

The DPL table is located in the labeled common block /XDPL/.

2.4.1.5 CEITBL (Control Entry Information Table)

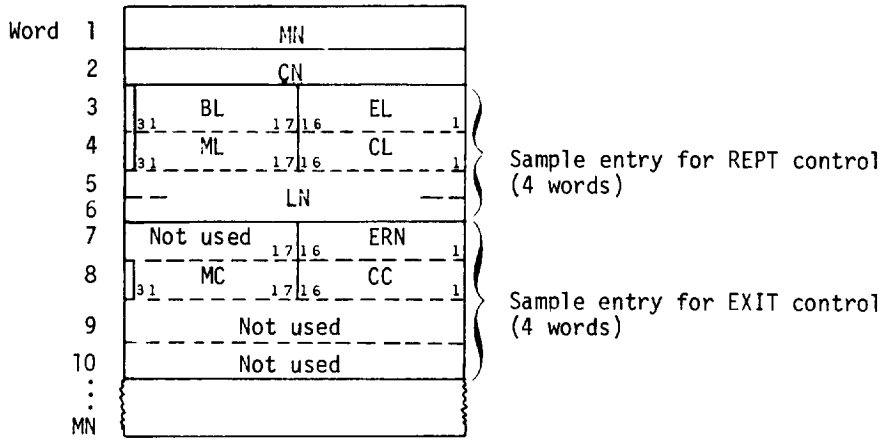
Description

CEITBL controls the REPT and EXIT DMAP module execution.

Created In Module

XGPI.

Table Format



Word	Item	Description
1	MN	Maximum number of words in table (integer).
2	CN	Current number of words being used (integer).
3	BL EL	ØSCAR record number where loop begins (integer). ØSCAR record number where loop ends (integer).
4	ML CL	Maximum loop count as specified in REPT instruction (integer). Current loop count, that is, the number of times loop has been repeated (integer).
5,6	LN	Location name specified in REPT instruction (BCD).
7	ERN	EXIT ØSCAR record number (integer).
8	MC CC	Maximum count specified in EXIT instruction. Current count of number of times EXIT instruction not executed.
9,10		These two words are zeroed.

Notes

CEITBL is located in named common block /XCEITB/.

2.4.1.6 VPS (Variable Parameter Set Table)

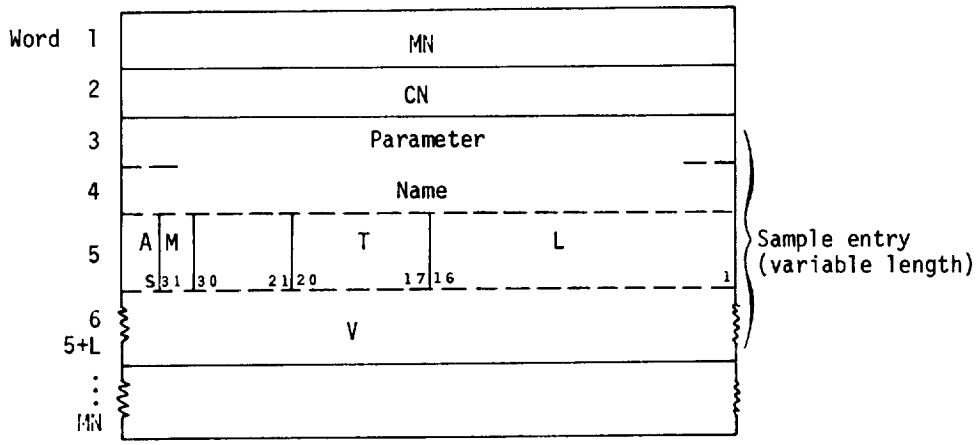
Description

The VPS table contains the values of all variable parameters referenced by DMAP modules in a DMAP program. It is the means for transferring variable parameter values from one module to another.

Created in Module

XGPI.

Table Format



Word	Item	Description
1	MN	Maximum number of words in VPS (integer).
2	CN	Current number of words being used (integer).
3,4	Name	BCD name of variable parameter.
5	A	Assigned flag. A = 1 indicates value from DMAP instruction has been entered in VPS.
	M	Modified flag. M = 1 indicates parameter was modified by bulk data PARAM card on restart.
	T	Type code for parameter (integer).
	L	Length in words of item V (integer).
6 thru 5+L	V	Value of parameter.

Notes

1. Items A, M and T (word 5 of sample entry) are used only by the XGPI module and are cleared prior to exiting XGPI.

2. Type code and corresponding word length.

T	L
1 = integer	1
2 = real, S.P.	1
3 = BCD	2
4 = real, D.P.	2
5 = complex, S.P.	2
6 = complex, D.P.	4

3. The VPS table is located in the named common block /XVPS/.

2.4.1.7 TAPID (Problem Tape Identification Table)

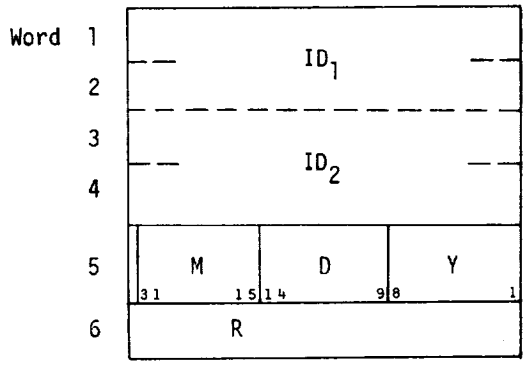
Description

TAPID contains Problem Tape identification information.

Created in Module

XCSA.

Table Format



Word	Item	Description
1,2	ID_1	First BCD field on ID Executive Control card.
3,4	ID_2	Second BCD field on ID Executive Control card.
5	M,D,Y	The date - integers - month, day, year.
6	R	Reel number of Problem Tape (integer).

Notes

1. TAPID is written on Problem Tape as single record field. It is always the first file on the Problem Tape.
2. ØTAPID has same format as TAPID. ØTAPID is the ID information from an Old Problem Tape being used for restarting problem.
3. TAPID and ØTAPID are located in named common block /STAPID/.

2.4.1.8 SYSTEM (NASTRAN System Parameters).

Description

A NASTRAN resident memory table containing various machine dependent, operating system and NASTRAN parameters. The length of the table is defined by one of the table items.

Created in Module

Most items are initialized by the NASTRAN block data program, SEMDBD. Several machine dependent items are set by subroutine BTSTRP.

Table Format

The sequential table description follows:

Word	Symbol	Description	Initially Set by
1	SYSBUF	Number of words in a GINØ buffer	BTSTRP
2	ØUTTAP	FØRTRAN logical unit for output	BTSTRP
3	NØGØ	Flag defining status during Preface	SEMDBD
4	INTP	FØRTRAN logical unit for input	BTSTRP
5	MPC	Multipoint constraint set ID	SEMDBD
6	SPC	Single-point constraint set ID	SEMDBD
7	METHØD	Eigenvalue extraction method	SEMDBD
8	LØAD	First record pointer in Case Control data block	SEMDBD
9	NLPP	Number of lines per page of printed output	BTSTRP
10	MTEMP	Material temperature set ID	SEMDBD
11	NPAGES	Current page count	SEMDBD
12	NLINES	Number of lines on current page	SEMDBD
13	TLINES	Total number of lines printed for problem	SEMDBD
14	MXLINS	Maximum number of printed lines permitted	SEMDBD
15	DATE(1)	Today's date - integer month 1-12	SEMDBD
16	DATE(2)	Today's date - integer day 1-31	SEMDBD
17	DATE(3)	Today's date - integer year (XX)	SEMDBD
18	TIMEZ	Time of problem start - seconds after midnite	SEMDBD
19	ECHØF	Bulk data output request type	SEMDBD
20	PLØTF	Structural plot request flag	SEMDBD
21	APPRCH	Approach type flag (1 = FØRCE, 2 = DISPL, 3 = DMAP)	SEMDBD
22	MACH	Computing machine code number (1 = 7094, 2 = 360, 3 = 1108, 4 = 6600)	BTSTRP
23	LSYSTM	Length of this table	SEMDBD
24	EDTUMF	User master file edit flag	SEMDBD
25	SWITCH	Logical sense switch bits set by a DIAG Executive Control Deck card	SEMDBD
26	CPPGCT	XCHK module page count	SEMDBD
27	MN	Used only in a conical shell problem. The lower order 16 bits contain N, the number of harmonics; the next higher order 16 bits contain M, the number of rings.	SEMDBD
28	ICØNFG	Machine configuration - subset of MACH code number	SEMDBD
29	MAXFIL	Maximum number of files to be added to FIAT	SEMDBD
30	MAXØPN	Maximum number of files to be opened simultaneously	SEMDBD
31	KØN360	Number of memory words to be released for OS (360 only)	SEMDBD
32	TIMEW	Initial problem start time (integer seconds after midnite)	SEMDBD

Word	Symbol	Description	Initially Set by
33	ØFPFLG	ØFP operate flag - set nonzero when ØFP operates	SEMDBD
34	NBRCBU	Length of FET + circular buffer (CDC 6600 only)	SEMDBD
35	NBRMST	Length of master index (CDC 6600 only)	SEMDBD
36	NBRSUB	Length of subindex (CDC 6600 only)	SEMDBD
37-38		Not used	
39	NBPC	Number of bits per character	BTSTRP
40	NBPW	Number of bits per word	BTSTRP
41	NCPW	Number of characters per word	BTSTRP

2.4.1.9 XLINK (Link Specification Table - Non-resident Edit)

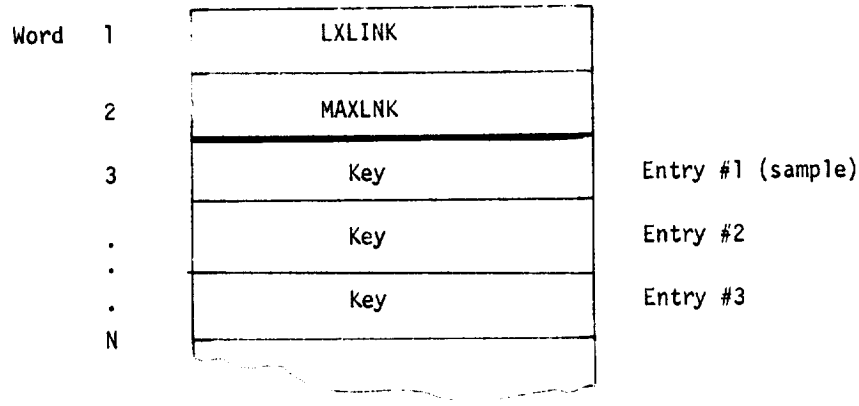
Description

This Link Specification Table (see also 2.4.2.7) contains an entry corresponding to each DMAP module within the MPL (2.4.2.2) table. These entries are indexed by MPL position and are thus ordered the same as the MPL entries. Each entry contains a key indicating the links in which the module resides.

Created in Module

XLINK data is created from the LNKSPC (2.4.2.7) and MPL (2.4.2.2) tables by the XGPIBS subroutine within the XGPI (4.7) module.

Table Format



Word	Item	Description
1	LXLINK	Length of table (number of entries)
2	MAXLNK	Maximum permissable link number
3,N	Key	Link residence key for the corresponding MPL entry

The content of this Key word is identical to the Key word within LNKSPC (2.4.2.7) for the machine type currently operating. See section 2.4.2.7 for an explanation of the content.

Notes

1. The XLINK table must contain an entry in the same order for each module that is in the MPL (2.4.2.2) table.
2. XLINK table is located in /XLINK/.

2.4.2 Executive Tables Not Permanently Core Resident

2.4.2.1 ØSCAR (Operation Sequence Control Array)

Description

The Operation Sequence Control Array (ØSCAR) controls the sequence of modules executed and aids in communicating data between modules.

The ØSCAR is generated from a DMAP instruction sequence supplied by the user or selected from the Rigid Format library. In general, an ØSCAR entry is a DMAP statement which has been translated to a more readily usable form for internal use.

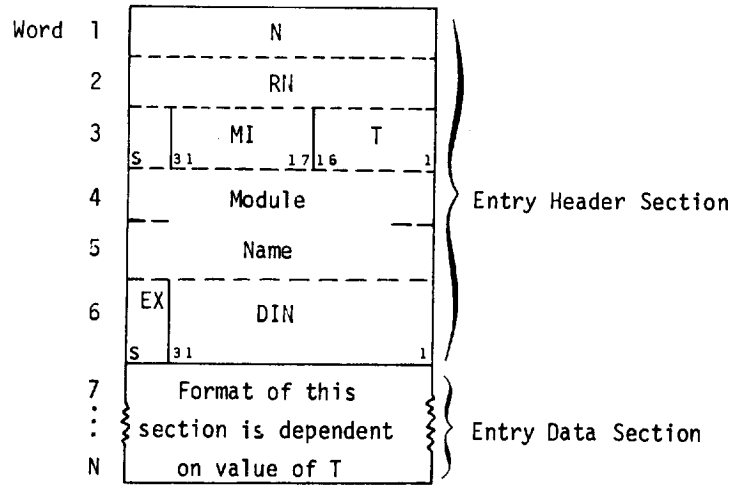
The four ØSCAR entry format types are:

1. Type 1 or F (functional) format is used for all functional modules except output processors.
2. Type 2 or Ø (output) format is used for output processors.
3. Type 3 or C (control) format is used for REPT, JUMP, CØND and END DMAP instructions.
4. Type 4 or E (executive) format is used for SAVE, CHKPNT, PURGE and EQUIV DMAP instructions.

Created in Module

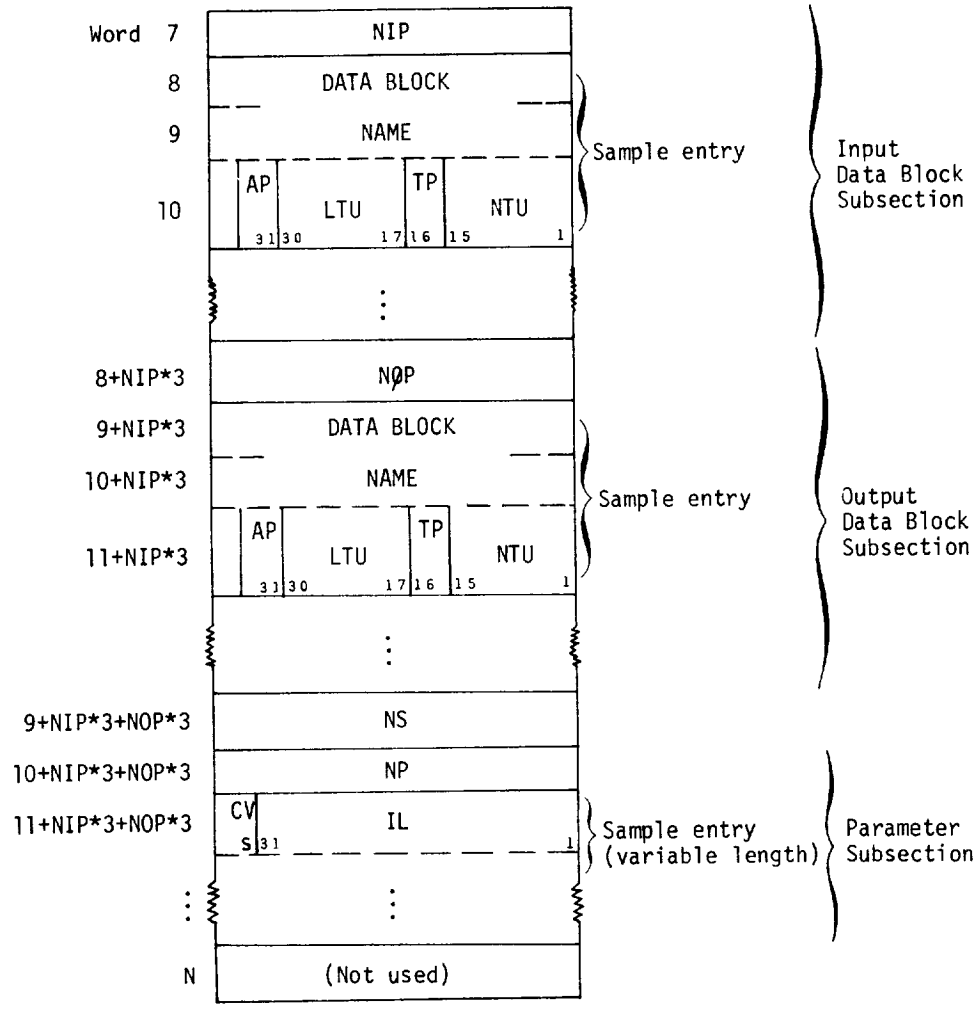
XGPI.

Table Format



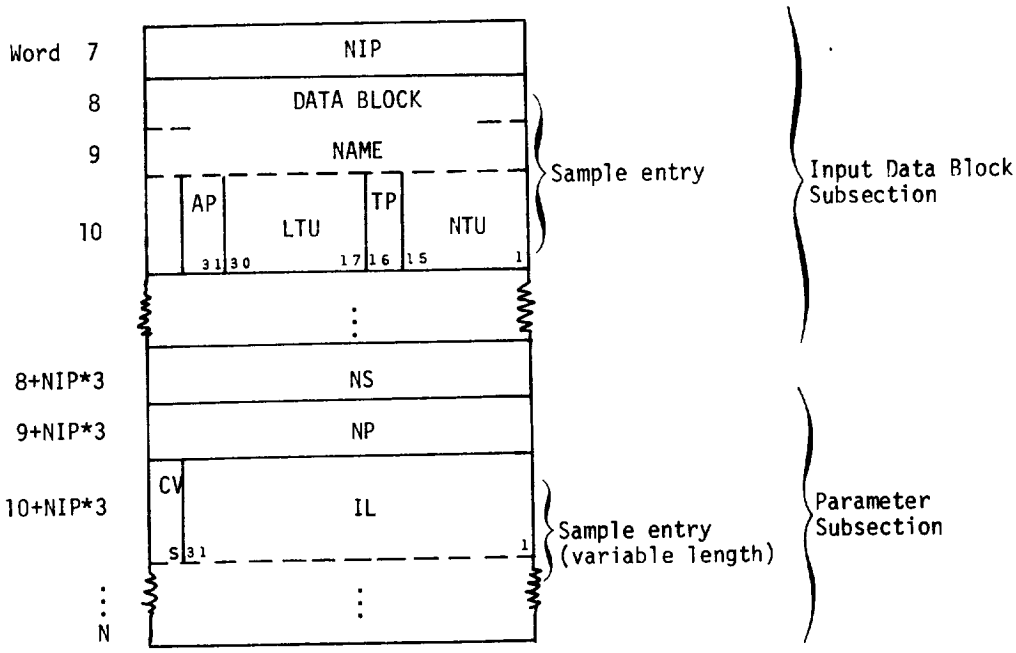
Word	Item	Description
1	N	Integer - number of words in entry.
2	RN	Integer - record number of entry in ØSCAR table.
3	MI	Integer - module index number assigned according to module's position in MPL and used to access the module's link specifications in /XLINK/.
	T	Integer - format type code (1, 2, 3, or 4) for data section of entry.
4,5	Name	BCD - module name is same as DMAP instruction name except when T = 4.
6	EX	Execute flag. EX = 1 indicates module is to be executed.
	DIN	Integer - DMAP instruction number which generated this entry.

Data Section Format for Type 1 or F Format:



Word	Item	Description
7	NIP	Integer - number of data blocks input to module as specified in MPL.
8,9	NAME	BCD - name of first input data block specified in DMAP instruction or zero if none specified.
10	AP	Append flag used by subroutine XSFA and set by XØSGEN if APPEND is specified for data block in a FILE DMAP instruction.
	LTU	Integer - last time used. ØSCAR record number of entry after which data block will no longer be saved.
	TP	Tape flag used by subroutine XSFA and set by XØSGEN if tape is specified for data block in a FILE DMAP instruction.
	NTU	Integer - next time used. ØSCAR record number of entry where data block is next referenced.
8 + NIP*3	NØP	Integer - number of data blocks output from module as specified in MPL.
9 + NIP*3, 10 + NIP*3	NAME	BCD - name of first output data block specified in DMAP instruction or zero if none specified.
11 + NIP*3	AP,LTU, TP,NTU	Same descriptions as word 10.
9 + NIP*3 + NØP*3	NS	Integer - number of scratch data blocks used by module as specified in MPL.
10 + NIP*3 + NØP*3	NP	Integer - number of parameters used by module as specified in MPL.
11 + NIP*3 + NØP*3	CV	Constant/variable flag. Flag indicates meaning of IL.
	IL	Integer - VPS index/length of constant. If CV = 0 the parameter is a constant whose value is stored in the next IL words (i.e., words 12 + NIP*3 + NØP*3 through 11 + NIP*3 + NØP*3 + IL). If CV = 1 the parameter is a variable whose value is stored in the VPS table. IL points to the value in VPS.

Data Section Format for Type 2 or 0 Format:

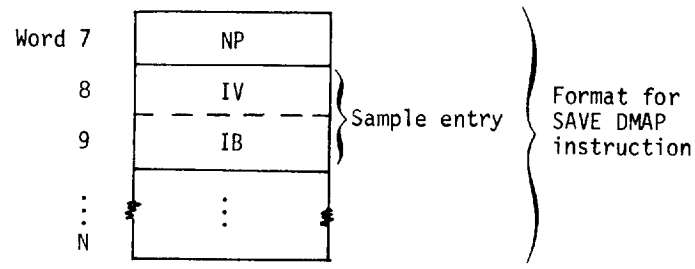


Type 1 format description is applicable to type 2 format above.

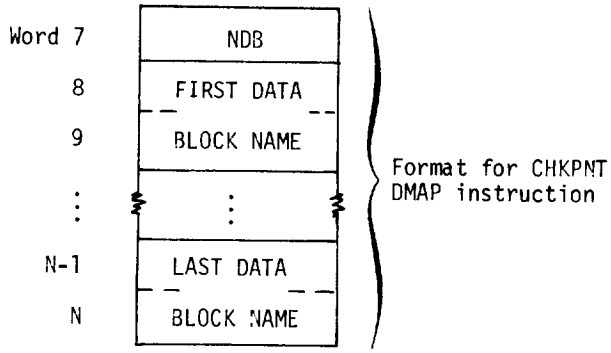
Data Section Format for Type 3 or C Format:

Word	Item	Description
N	RRN	Integer - re-entry record number. Indicates ØSCAR record to jump to for JUMP, REPT and CØND DMAP instructions. Not applicable for EXIT so RRN = 0.
	I	Integer - index into XCEITBL for REPT or EXIT DMAP instruction. Pointer to parameter value in XVPS table if CØND DMAP instruction. Not applicable for JUMP so I = 0.

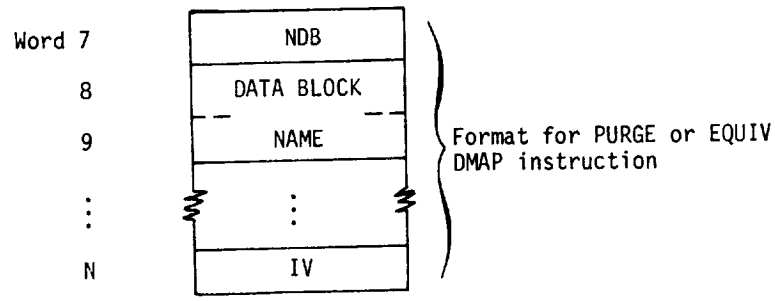
Data Section Formats for Type 4 or E Formats:



Word	Item	Description
7	NP	Integer - number of parameter values to be saved (i.e. number of entries).
8	IV	Integer - pointer to parameter value in VPS.
9	IB	Integer - pointer to parameter value in blank common.



Word	Item	Description
7	NDB	Integer - number of data blocks names in list.
8 thru N	NAMES	BCD - list of data blocks to be checkpointed.



Word	Item	Description
7	NDB	Integer - number of data block names in first group. There may be one or more groups.
8,9	NAME	BCD - name of first data block in first group.
N	IV	Integer - pointer to parameter value in VPS table.

Notes

1. ØSCAR is located in named common block /XGPI1/ while module XGPI is generating it.
2. After generating ØSCAR and prior to exiting XGPI the ØSCAR is written on the Data Pool File. The ØSCAR file header ID is XØSCAR.

2.4.2.2 MPL (Module Property List)

Description

The Module Properties List contains information needed by the module XGPI to correctly generate ØSCAR table entries for executable DMAP instructions and/or to determine whether or not the DMAP instructions adhere to the calling sequence described in section 4, Module Functional Descriptions.

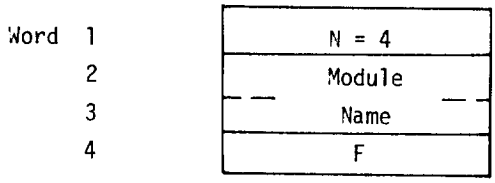
Created in Module

XGPI (Block Data Program XMPLBD).

Table Format

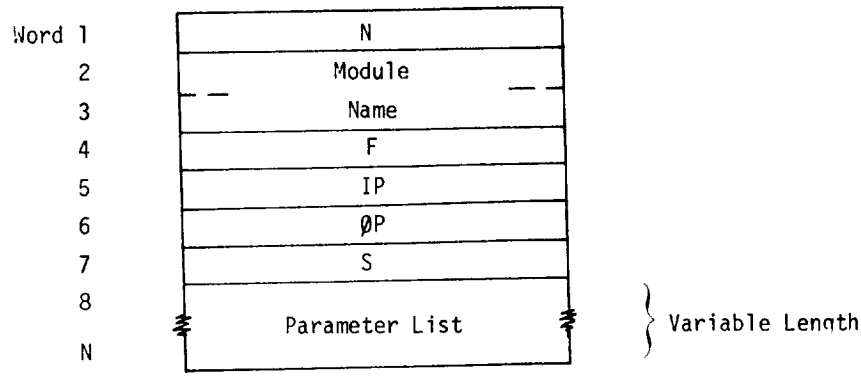
There are two formats used in the MPL, one for Declarative (FILE, BEGIN, LABEL), Executive (CHKPNT, EQUIV, PURGE, SAVE) and Control (REPT, JUMP, CØND, EXIT, END) DMAP modules and the other for functional modules. All entries in the MPL are integer except for module names which are BCD and BCD parameter values.

Format for Declarative, Executive and Control Modules:



Word	Item	Description
1	N	Number of words in entry.
2,3	Name	Name of DMAP module.
4	F	ØSCAR format type code 3 = Control module (C format) 4 = Executive module (E format) 5 = Declarative module (D format)

Format for Functional Modules:



Word	Item	Description
1-3		Same as format for Declarative, Executive and Control modules.
4	F	F = 1 implies module has input and output data blocks F = 2 implies module has no output data blocks, e.g., ØFP, SETVAL etc.
5	IP	Number of input data blocks.
6	ØP	Number of output data blocks.
7	S	Number of scratch data blocks.
8-N		Parameter List (omit if no parameters).

The parameter list for a module contains the types of all parameters residing in blank common that are referenced by the module. The order of the parameters in the MPL entry must coincide with the order of the parameters in blank common. Space must be allowed for a default value if the parameter type code is positive. The space following a positive type code will contain the default value if the type code is integer or BCD, otherwise the space will contain an index into another table which contains the default value.

Type Code	Space Needed for Default Values
1 = integer	1 word
2 = real, single precision	1 word
3 = BCD	2 words
4 = real, double precision	2 words
5 = complex, single precision	2 words
6 = complex, double precision	4 words

An example of all possible entries in a parameter list follows. Note that for each parameter only the first index word will appear in the XMPLBD Block Data subprogram.

	Word		
1	8	=	Integer type code.
-3	9	=	Integer default value.
-1	10	=	Integer type code (no default value).
2	11	=	Real, S.P. type code.
1	12	=	Index into table containing a real S.P. default value.
-2	13	=	Real, S.P. type code (no default value).
3	14	=	BCD type code.
ABCD	15	=	BCD default value (2 words).
EFGH	16		
-3	17	=	BCD type code (no default value).
4	18	=	Real, D.P. type code.
1	19	=	Index into table containing a real D.P. default value.
1	20		Note index is in both words.
-4	21	=	Real, D.P. type code (no default value).
5	22	=	Complex, S.P. type code.
2	23	=	Index into table containing a complex S.P. default value.
2	24		
-5	25	=	Complex, S.P. type code (no default value).
6	26	=	Complex, D.P. type code.
3	27	=	Index into table containing the real part of the complex D.P. default value.
3	28		
4	29	=	Index into table containing the imaginary part of the complex D.P. default value.
4	30		
-6	31	=	Complex, D.P. type code (no default value).

Notes

1. MPL table is located in named common block /XGPI2/.
2. The default value table is located in named common block /XGPI2X/.

2.4.2.3 XPTDIC (Problem Tape Dictionary)

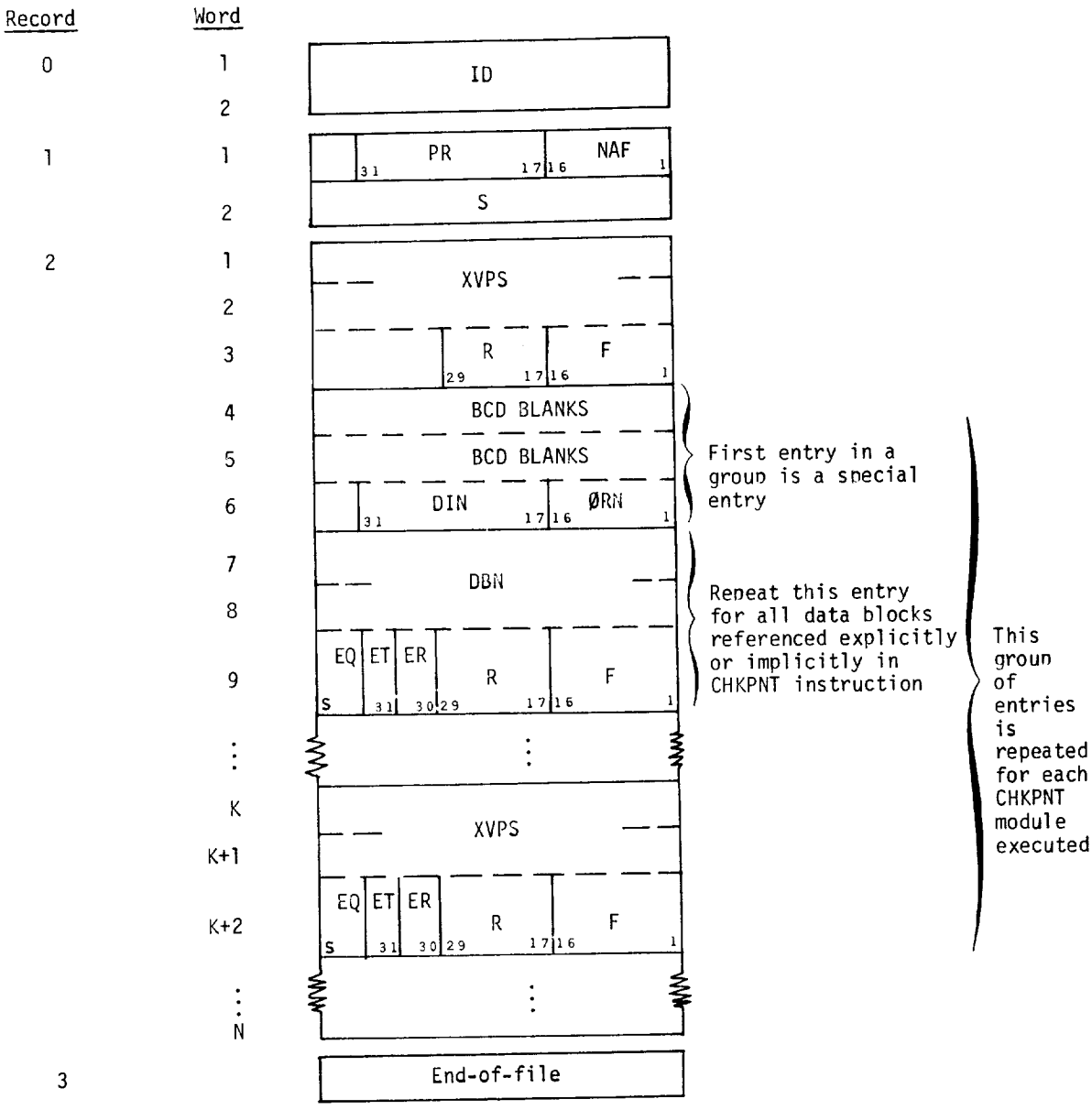
Description

XPTDIC is the Problem Tape Dictionary of data blocks checkpointed plus other information needed to restart a problem.

Created in Modules

XGPI, CHKPNT and XCEI.

Table Format



Record	Word	Item	Description
0	1,2	ID	Header record containing name XPTDIC (BCD).
1	1	PR	Present reel number of Problem Tape. Reels are numbered sequentially beginning with Reel 1.
		NAF	Next available file number on present reel. Files are numbered sequentially beginning with file 1.
	2	S	Sequence number of last restart dictionary card punched out.
2	1,2	XVPS	BCD name XVPS. The file corresponding to this entry contains named common blocks /XVPS/ and /XCEITB/.
	3	R,F	Reel number and file number where the file corresponding to this entry is located. For this entry the reel number must be one.
	4,5	(blanks)	BCD blanks indicate special entry.
	6	DIN	DMAP instruction number of DMAP instruction following CHKPNT module (i.e., re-entry point).
		ØRN	ØSCAR record number of CHKPNT module being executed.
	7,8	DBN	Data block name (BCD) of data block being checkpointed.
	9	EQ	Equivalence flag. EQ = 1 indicates data block is equivalenced to another data block.
		ET	End of tape flag. ET = 1 indicates that data block is split across two reels of problem tape.
		ER	End of logical record flag. ER = 1 indicates that the complete logical record was written out prior to changing reels when ET = 1.
		R,F	Reel number and file number where the file corresponding to this entry is located. For purged or not-generated data blocks, R = 0 and F = 0.
	K, K+1	XVPS	BCD name XVPS. The file corresponding to this entry contains named common blocks /XVPS/, /XCEITB/ and /SYSTEM/.
	K+2	EQ,ET, ER,R,F	See word 9 for descriptions.

Notes

1. All entries are integer unless otherwise noted.
2. The XPTDIC table is always the last file on the Problem Tape.
3. XGPI generates records 0, 1 and the first entry of record 2. CHKPNT modules add entries to record 3. XCEI drops entries from record 2 when a REPT DMAP instruction transfers control to the top of a DMAP loop.
4. XCSA also creates a XPTDIC table when problem is being restarted. This special XPTDIC table is created from the restart dictionary and its format is essentially the same as described above except that there are no special entries.

2.4.2.4 PVT (Parameter Value Table)

Description

The Parameter Value Table contains the parameter names and values of all parameters input by means of the PARAM bulk data card.

Created in Module

IFP.

Table Format

Record	Word	Item	
0	1,2	Header record contains name PVT (BCD).	
1	1,2	Name of parameter (BCD)	repeat for all parameters on PARAM cards.
	3	Type code for parameter value	
	4	Value of parameter. Type codes and corresponding lengths, L, of values are given in table below.	
	:		
	3+L		
	:		

Notes

Type Code	Meaning of Code	Corresponding Length in Words
1	Integer	1
2	Real, single precision	1
3	BCD	2
4	Real, double precision	2
5	Complex, single precision	2
6	Complex, double precision	4

1. IFP does not create PVT if no PARAM cards exist in the Bulk Data Deck.
2. PVT is written on the Problem Tape as 2 or more records (a header record and 1 record for each PARAM card).
3. The PVT table is located in named common block /XPVT/.

2.4.2.5 XCSA (Executive Control Table)

Description

Executive control table derived from the Executive Control Deck.

Created in Module

XCSA.

Table Format

Record	Word	Item	
0	1	BCD word XCSA - header ID.	
	2,3	BCD word SØL } Dictionary of contents of records	
	4,5	BCD word DMAP } to follow. Does not need to be in	
	6,7	BCD word MED } this order, nor is MED always present.	
1	1	Approach code	SØL record
	2	Start code	
	3,4	Alter parameters	
	5	Solution number	
	6	Subset number	
2	1	RD table (packed DMAP program)	DMAP record (BCD information)
	:	or user generated DMAP	
	N	program (18 words per card image).	
3	1	Number of DMAP instructions	MED record included only if approach calls for a Rigid Format
	2	Number of words per ISl table entry.	
	3		
	:	ISl table (Module Execution Decision Table).	
	L	Number of entries in JNM table.	
	L+1		
	:	JNM table (File Name Table)	
	M	Number of entries in INM table	
	M+1		
	:	INM table (Card Name Table)	
4		End-of-file	

Notes

1. Data block XCSA is written on the Problem Tape.
2. A more detailed description of tables ISl, JNM and INM is given in the Module Functional Description for module XCSA, section 4.2.

2.4.2.6 XALTER (Executive Alter Table)

Description

XALTER is generated from the ALTER data in the Executive Control Deck.

Created in Module

XCSA.

Table Format

Record	Word	Item		
0	1,2	BCD word XALTER - header record		
1	1,2	Numbers of DMAP instructions to be altered. (Integers).		
2	1	Card image (BCD)	Zero or more of these records. Repeat until next 2 word record encountered.	Repeat 1 or more times until EØF encountered.
	⋮			
	18			
⋮				
N		End-of-file		

Notes

XALTER data block is written on the Problem Tape.

2.4.2.7 LNKSPC (Link Specification Table - Resident Base)

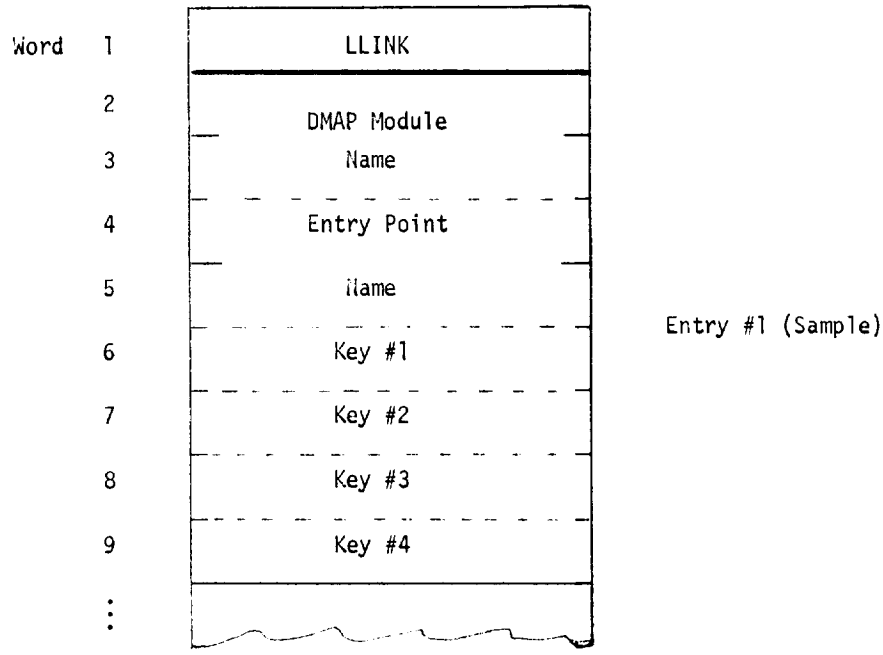
Description

This Link Specification Table (see also 2.4.1.9) contains an entry for each executable DMAP module available within the NASTRAN system. Each entry contains: a) the DMAP module name, b) the module's subroutine entry point name and c) a key indicating the links in which the module resides for each of four machine types.

Created in Module

LNKSPC data is stored by the XBSBD Block Data subprogram in module XGPI (4.7).

Table Format



Word	Item	Description
1	LLINK	Length of table in words (excluding word 1)
2,3	DMAP Module Name	DMAP name-8 characters (4 characters/word)
4,5	Entry Point Name	Entry name-8 characters (4 characters/word)
6	Key #1	Link residence key for machine type #1
7	Key #2	Link residence key for machine type #2
8	Key #3	Link residence key for machine type #3
9	Key #4	Link residence key for machine type #4

The machine type code number is the same as that defined in the MACH word of the SYSTEM (2.4.1.8) table. Each bit within the Key word specifies a link number that is to contain that module. Bits are numbered from right to left; the right most (least significant) bit specifies that the module is to reside in link 1, etc. For example, if a particular Key contained 26 (binary 011010), only links 2, 4 and 5 would contain the specified module.

Notes

1. The LNKSPC table must contain an entry for each executable module that is in the MPL (2.4.2.2) table.

2. The LNKSPC table is located in /XLKSPC/.

2.5 MISCELLANEOUS TABLE DESCRIPTIONS.

The following is an alphabetical index of miscellaneous table descriptions.

Section Number	Table Name	Where Stored	Page Number
2.5.2.2	BITPØS	/BITPØS/	2.5-7
2.5.2.4	CHAR94	/CHAR94/	2.5-10
2.5.2.7	CHRDRW	/CHRDRW/	2.5-13
2.5.1.6	DESCRP	/DESCRP/	2.5-3
2.5.2.1	GPTA1	/GPTA1/	2.5-6
2.5.1.5	MSGX	/MSGX/	2.5-3
2.5.1.8	NAMES	/NAMES/	2.5-4
2.5.1.1	ØSCENT	/ØSCENT/	2.5-2
2.5.1.2	ØUTPUT	/ØUTPUT/	2.5-2
2.5.2.3	PLTDAT	/PLTDAT/	2.5-8
2.5.1.3	STIME	/STIME/	2.5-2
2.5.2.6	SYMBLS	/SYMBLS/	2.5-12
2.5.1.7	TWØ	/TWØ/	2.5-4
2.5.1.9	TYPE	/TYPE/	2.5-4
2.5.1.4	XMDMSK	/XMDMSK/	2.5-3
2.5.2.5	XXPARM	/XXPARM/	2.5-11

2.5.1 Miscellaneous Tables Which Are Permanently Core Resident.

2.5.1.1 ØSCENT (ØSCAR Entry)

Description

A 200 word storage array containing the ØSCAR entry (record) currently being processed.

Created in Module

The entry is read from the ØSCAR and stored in ØSCENT by the XSEMi (section 3.3.7) subroutine. Other executive routines that require details of the current entry will search ØSCENT.

Table Format

The ØSCENT format is identical to the ØSCAR (section 2.4.2.1) entry it currently contains.

2.5.1.2 ØUTPUT (Output headings)

Description

A storage array containing problem title, subtitle, label and various headings required by the PAGE (section 3.4.24) routine to properly annotate the NASTRAN output.

Created in Module

The title, subtitle and label are taken from Case Control Deck cards and stored in ØUTPUT by IFP1 (section 4.3). Other heading lines may be stored by output modules prior to calling PAGE.

Table Format

ØUTPUT contains sufficient space for seven 128 character lines. The first three lines contain the title, subtitle, and label. The subsequent three lines contain local headings, and the final line contains the plotter ID. Since 4 characters occupy each computer word, the ØUTPUT array requires 224 words of storage.

2.5.1.3 STIME (Solution Time)

Description

A storage cell containg the user's estimated solution time.

Created in Module

The estimated solution time is taken from a Executive Control Deck card and stored into STIME by XCSA (section 4.2)

Table Format

STIME consists of a single cell containing the estimated time in integer seconds.

2.5.1.4 XMDMSK (Executive Module Decision Mask)

Description

Contains the 155 bit master module execution mask (see section 1.10) and a cell indicating checkpoint status.

Created in Module

The 155 bit master module execution mask is generated and used by XGPI (section 4.7). The checkpoint status set on (YES) by XCSA (section 4.2) by the presence of a CHKPNT = YES card in the Executive Control Deck.

Table Format

The 155 bit mask occupies the low order 31 bits of the first five words of XMDMSK. The sixth word is the checkpoint status (flag).

2.5.1.5 MSGX (Message Queue)

Description

A queue table to hold four word NASTRAN information and error messages between the time they are generated by a module and printed by the message writer, MSGWRT (section 3.4.26).

Created in Module

Messages may be generated by any NASTRAN module through a call to MESAGE (section 3.4.25).

Table Format

Word 1 - Number of messages queued.
Word 2 - Maximum number of messages queue can hold
Word 3-6 - Four word message entry (typical)
Word 6-end - Additional four word message entries

2.5.1.6 DESCRP (Matrix Description)

Description

A storage block used by subroutine INTPK (section 3.5.3) to buffer the matrix unpacking procedure. This buffering reduces the number of I/Ø accesses to the particular matrix data block.

Created in Module

DESCRP is filled and used exclusively by INTPK

Table Format

An array with the first word defining the length of the array.

2.5.1.7 TWØ (Powers of Two)

Description

A 32 word array with each word (starting with 1 in the 32nd word) containing the next power of two.

Created in Module

The 32 integer values are defined within the NASTRAN system block data program (SEMDBD).

Table Format

Word 32 - 1
Word 31 - 2
Word 30 - 4
Word 29 - 8
etc.

2.5.1.8 NAMES (Symbolic Names)

Description

A series of symbolic names identified with their NASTRAN numeric equivalents. Defines values for GINØ file options, arithmetic types and matrix forms.

Created in Module

The values are defined within the NASTRAN system block data program (SEMDBD).

Table Format

Word	SYMBØL	VALUE	Word	SYMBØL	VALUE	Word	SYMBØL	VALUE
1	RD	= 2	7	EØFNRW	= 3	13	RECT	= 2
2	RDREW	= 0	8	RSP	= 1	14	DIAG	= 3
3	WRT	= 3	9	RDP	= 2	15	UPPER	= 4
4	WRTREW	= 1	10	CSP	= 3	16	LOWER	= 5
5	REW	= 1	11	CDP	= 4	17	SYM	= 6
6	NØREW	= 2	12	SQUARE	= 1	18	RØW	= 7
						19	IDENT	= 8

2.5.1.9 TYPE (Number Types)

Description

A series of properties are defined as a function of a number type. The type may be Real Single Precision (RSP-1), Real Double Precision (RDP-2), Complex Single Precision (CSP-3), or Complex Double Precision (CDP-4). The properties that may be returned include precision (single, double), number of words, and real or complex.

Created in Module

The properties are defined within the NASTRAN system block data program (SEMDBD).

Table Format

Word	Property (Values)	Type		
1	1	Precision (RSP)		
2	2	Precision (RDP)		
3	1	Precision (CSP)	Words (RSP)	
4	2	Precision (CDP)	Words (RDP)	
5	2		Words (CSP)	
6	4		Words (CDP)	
7	1			Real/Complex (RSP)
8	1			(RDP)
9	2			(CSP)
10	2			(CDP)

Example

Assume the number of words required to contain a Complex Single Precision (CSP-3) is desired. The third item in the Words array is indexed and found to contain a 2 (words).

2.5.2 Miscellaneous Tables Not Permanently Core Resident.

2.5.2.1 GPTA1

Purpose

To describe connection and property characteristics of each element. GPTA1 is used in modules GP1, GP2, GP3 and TA1, and is initialized by the block data program GPTABD.

Description

Word	Type	Item	
1	I	Number of entries (i.e., elements) in table	
2	I	Pointer to first word of last entry in table	
3	I	Number of words per entry in table	
4-5	B	Name of element as it appears on connection card (e.g., CRØD)	repeated for each element
6	I	Internal element identification number	
7-8	I	ECT record ID and trailer bit for LØCATE	
9	I	Number of words per entry on ECT	
10-11	I	EPT record ID and trailer bit for LØCATE	
12	I	Number of words per entry on EPT	
13	I	Number of grid points per element	
14	I	+1 : Scalar element with grid point and component code 0 : Not a scalar element -1 : Scalar element with scalar points only	
15	I	Number of words per entry on EST	
16	I	Position of first grid point in ECT entry	

2.5.2.2 BITPØS.

Purpose

To provide pointers into USET and USETD words for interpreting the nested vector sets in NASTRAN.

Description

Word	Item	
1	UM	Bit number
2	UO	Bit number
3	UR	Bit number
4	USG	Bit number
5	USB	Bit number
6	UL	Bit number
7	UA	Bit number
8	UF	Bit number
9	US	Bit number
10	UN	Bit number
11	UG	Bit number
12	UE	Bit number
13	UP	Bit number
14	UNE	Bit number
15	UFE	Bit number
16	UD	Bit number

Note

All words are integer.

2.5.2.3 PLTDAT

Purpose

To define plotter-dependent parameters.

Description

This table is defined in the PLØTBD block data subprogram. The table is divided into N+2 20-word sections, where N = number of plotters acceptable by the NASTRAN plotting software. Sections 3 to N+2 are the only sections initialized, because each contains values which are dependent upon the plotter hardware. Section 1 contains values which may vary within the limits of the hardware, and Section 2 is simply a duplicate of one of the last N sections corresponding to the plotter of interest.

Section 2 must be filled in by the module writer. The format of Sections 2 to N+2 is as follows:

Word	Type	Name	Description
1-2	R	XYMAX	Maximum x and y coordinate values acceptable by the plotter.
3	R	CNTSIN	Number of plotter counts/inch on paper.
4-5	R	CNTCHR	Number of plotter counts per character in the x and y directions.
6	R	MAXLEN	Maximum length of a line segment.
7	I	NPENS	Maximum number of pens or line density available on the plotter.
8-9	R	ØRIGIN	For incremental plotters, the current pen position relative to the lower left corner of the plot. Otherwise, the location of the lower left corner of the plotter relative to its true physical origin.
10	I	PLTYPE	+1, +2, or +3 if the plotter is a microfilm, table or drum plotter, respectively, with typing capability. -1, -2, or -3 if the plotter is a microfilm, table or drum plotter, respectively, with no typing capability (i.e., all characters must be drawn).
11	B	PLTAPE	PLT1 if an even parity plot tape is to be generated for this plotter. PLT2 if an odd parity plot tape is to be generated for this plotter.
12	I	PBFSIZ	Plot tape physical record size (number of characters)
13	I	EØF	0 if an end-of-file is to be written after every plot. 1 if no end-of-file is to be written on the plot tape.
14-20			Undefined

Section 1 must also be filled in by the module writer. However, unlike Section 2, some of the parameters may vary from plot to plot, as long as they remain within the limitations imposed by the plotter hardware. The format of Section 1 is as follows.

Word	Type	Name	Description
1	I	MØDEL	Plotter model index
2	I	PLØTER	Plotter index
3-6	R	REG	Plotter region (x_{min}, y_{min}, x_{max}, y_{max}) in which the current picture is being drawn. These values must be some fraction (between 0 and 1) of words 7 and 8 (AXYMAX).
7-8	R	AXYMAX	Size of the paper used (x,y), less the borders, in plotter units.
9-10	R	XYEDGE	Size of the borders (x,y) in plotter units.
11	I	CAMNUM	Current selected camera. This word need not be filled in, because it is set and used as a communication between the SELCAM and SKPFRM subroutines.
12-20			Undefined

Usage

Sections 1 and 2 are normally setup by the FNDPLT subroutine, except for the plotter region values (REG). These values must be setup by the module writer himself. It is essential that both these sections be correctly setup, because they are referenced by the entire NASTRAN plotter software package.

If Sections 1 and 2 are correctly setup by the module writer, he need not subsequently worry about such things as compensating for paper margins or different physical plotter origins. He need only assume that the plotter origin is located at the lower left corner of the paper where the left and bottom borders intersect. The NASTRAN plotter software will automatically compensate for the borders and the physical origin.

2.5.2.4 CHAR94

Purpose

To provide a table of characters used to generate plot tapes as if the computer were always an IBM 7094. This table however is independent of the actual computer used.

Description

This is a 240 word table defined in the PLØTBD block data subprogram. It is divided into four equal sections of 60 words each. Each entry in each section has a parallel entry in the other three sections.

Section I is a string of all the Hollerith characters acceptable by the plot modules of the form 1Hx, where x is a Hollerith character.

Section II contains the integer equivalents of the IBM 7094 internal binary characters in the same order as Section I. However, near the end of this section are integers representing various additional characters not in Section I. These additional characters cannot be expressed in the form 1Hx and are used for special plotter commands. Each entry in this table is a right-adjusted two-digit integer with leading zeroes.

Section III contains the integer equivalents of the IBM 7094 BCD characters as they would appear on an even parity tape written on an IBM 7094, in the same order and form as in Section II.

Section IV contains the integer equivalents of the CDC display character codes so as to produce an even parity BCD plot tape as if written on an IBM 7094, in the same order and form as in Section II.

The sequence of characters in each section is as follows:

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	J
K	L	M	N	Ø	P	Q	R	S	T
U	V	W	X	Y	Z	(	)	+	-
*	/	=	.	,	$	'	b		

character 49 = end of record mark

character 50 = end of file mark

characters 51-53 = special characters.

characters 54-60 = 0

Note

In Section I, characters 49-60 = 0.

Usage

Section I is basically used for calculating an index into the other two sections by comparing an arbitrary Hollerith character with each character in Section I until a match is found. Once this is done, the index is used to extract the corresponding entry from either Section II or III, depending on whether an odd or even parity plot tape is being generated. If the computer is an IBM 7094, only Section II is used, and if the computer is a CDC 6600 and an even parity plot tape is being generated, Section IV is used instead of Section III.

2.5.2.5 XXPARM

Purpose

To define the plot tape buffer size, the camera to be used, the number of blank frames of film to be inserted between plots, the plotter model name, and the paper size to be used on table plotters.

Description

This table is defined as follows in the PLØTBD block data subprogram.

Word	Type	Name	Description
1	I	BUFSIZ	Plot tape buffer size
2	I	CAMERA	Plotter camera to be used (appropriate only on a microfilm plotter).
3	I	BFRAMS	Number of blank frames of film to be inserted between plots (appropriate only on a microfilm plotter).
4-5	I, or B	PLTMDL	Plotter model identification.
8-9	R	PAPSIZ	Width and height of the paper to be used (appropriate only on table plotters).

Usage

The initial values of these variables are as follows:

BUFSIZ = must be set by the module writer

CAMERA = 2 (paper output only)

BFRAMS = 1

PLTMDL = 4020, 0 (integer)

PAPSIZ = 8.5, 11.0

This table's actual size is 152 words. The remainder of the table is initialized for the structural plotter module, PLØT, but may be used by the programmer for anything he desires in other plotting modules.

2.5.2.6 SYMBLS

Purpose

To provide a table of irdices into the CHAR94 and CHRDRW tables used to type or draw predefined plotter symbols.

Description

The table is defined in the PLØTBD block data subprogram. There is room for up to 20 indices for each plotter. However, the same number of indices must be defined for each plotter. The format of the table is as follows:

Word	Type	Description
0	I	Number of symbols defined for each plotter (currently = 9).
1-20	I	Symbol indices for plotter 1.
21-40	I	Symbol indices for plotter 2.
41-60	I	Symbol indices for plotter 3.
⋮	⋮	⋮

There are as many groups of symbol indices as there are available plotters. The symbols defined for each plotter are as follows:

Symbol 1 = x
Symbol 2 = *
Symbol 3 = +
Symbol 4 = -
Symbol 5 = · (dot, not a period).
Symbol 6 = circle
Symbol 7 = square
Symbol 8 = diamond
Symbol 9 = triangle

Should any of these symbols not be available on a plotter, a substitution of another symbol must be made.

Usage

This table is used by the SYMBØL subroutine.

2.5.2.7 CHRDRW

Purpose

To define the combination of lines needed to draw alphanumeric characters and symbols.

Description

This table is defined in the PLØTBD block data subprogram. The table is divided into two sections. Section I is a list of indices into Section II, used to locate the data needed to draw characters. The first 48 indices in Section I correspond to the 48 characters listed in Section I of the CHAR94 table. The last 7 indices are used for drawing the special characters listed in the SYMBLS table. If an index is negative, it is an index into Section I instead of Section II. This occurs when a duplicate character exists (e.g., a zero, the letter "Ø", and the symbol for a circle).

Section II of this table defines the (integer) coordinates of the starting and ending points of the straight lines to be drawn in order to draw a character or symbol. In general, the necessary straight lines are contiguous, so that the end point of one line is the starting point of the next, etc. In some cases, this is either impractical or impossible (e.g., *, +, =, etc.). In such a situation, the starting point of a line is negative, meaning that it is not to be connected to the end point of the preceding line.

The characters defined in Section II are based upon 6x6 square characters. The values in this section are simply integer coordinates within a 6x6 square.

The format of this table is as follows:

Word	Type	Name	Description
0	I	LSTCHR	Name of characters and symbols referenced in Section I (=52).
1-60	I	CHRIND	Section I - "LSTCHR" indices into Section II.
61-760	I	CHR	Section II = (x,y) pairs defining the lineal representation of 6x6 square characters.

Usage

This table is used by the DRWCHR subroutine.

3.1 INTRODUCTION

Section 3 contains descriptions of subroutines not an integral part of a module. Those subroutines which are an integral part of a module are discussed in section 4, Module Functional Descriptions. Section 3.2 contains an alphabetical index of entry points of routines documented in section 3. A similar index of entry points documented in section 4 can be found in section 4.1.3.

Subroutine descriptions have been partitioned into 3 classifications: executive, utility and matrix subroutine descriptions, documented in sections 3.3, 3.4, and 3.5 respectively.

Descriptions of the plotting utility routines (e.g., AXIS, section 3.4.40; AXISi, section 3.4.41) refer to plotters by number or the letter "i", and to plotter models by number only. The correspondence of these numbers to plotter hardware is given in Table 1. Further details can be found in section 4 of the User's Manual.

Table 1. Correspondence Between External and Internal Plotter and Model Names and Numbers.

External plotter name	External model name	Internal plotter number	Internal model number
BL	STE,30	1	1
	LTE,30		
EAI	3500,30	2	1
	3500,45	8	1
SC	4020,0	3	1
CALCØMP	765,205	5	4
	765,210	4	4
	765,105	5	5
	765,110	4	5
	763,205	7	4
	763,210	6	4
	763,105	7	5
	763,110	6	5
	565,205	5	2
	565,210	4	2
	565,105	5	3
	565,110	4	3
	565,305	5	1
	565,310	4	1
	563,205	7	2
	563,210	6	2
	563,105	7	3
	563,110	6	3
	563,305	7	1
	563,310	6	1
DD	80,B	9	1
NASTRAN	M,0	10	+1
	T,0	11	+2
	D,0	11	+3
	M,1	10	-1
	T,1	11	-2
	D,1	11	-3

where:

BL = Benson Lehner
EAI = Electronic Associates Inc.
SC = Stromberg Carlson

Table 1. Correspondence Between External and Internal Plotter and Model Names and Numbers (Cont'd).

CALCØMP	= California Computing
DD	= Data Display
NASTRAN	= NASTRAN General Purpose Plotter

3.2 ALPHABETICAL INDEX OF ENTRY POINTS FOR SUBROUTINE DESCRIPTIONS.

Section Number	Entry Point	Subroutine Description	Page Number
3.5.10	ADD	ADD	3.5-19
3.4.1	ANDF	MAPFNS	3.4-1
3.4.40	AXIS	AXIS	3.4-70
3.4.41	AXIS3	AXISi	3.4-72
3.4.41	AXIS10	AXISi	3.4-72
3.4.7	BCKREC	BCKREC	3.4-9
3.3.5	BGNSYS	ENDSYS	3.3-6
3.5.1	BLDPK	BLDPK	3.5-1
3.5.1	BLDPKI	BLDPK	3.5-1
3.5.1	BLDPKN	BLDPK	3.5-1
3.3.2	BTSTRP	BTSTRP	3.3-2
3.5.5	CALCV	CALCV	3.5-12
3.5.16	CDCØMP	CDCØMP	3.5-62
3.4.4	CLØSE	CLØSE	3.4-5
3.4.18	CLSTAB	CLSTAB	3.4-26
3.4.1	CØMPLF	MAPFNS	3.4-1
3.5.16	CØM12	CDCØMP	3.5-63
3.3.12	CØNMSG	CØNMSG	3.3-16
3.4.1	CØRSZ	MAPFNS	3.4-1
3.4.1	CØRWDS	MAPFNS	3.4-1
3.5.16	CTRNSP	CDCØMP	3.5-63
3.5.15	DECØMP	DECØMP	3.5-44
3.5.15	DDLØØP	DECØMP	3.5-60
3.5.15	DLØØP	DECØMP	3.5-60
3.5.21	DMPY	DMPY	3.5-71
3.4.68	DRWCHR	DRWCHR	3.4-115
3.4.62	EJECT	EJECT	3.4-105
3.5.22	ELIM	ELIM	3.5-73
3.3.5	ENDSYS	ENDSYS	3.3-6
3.4.9	EØF	EØF	3.4-11

Section Number	Entry Point	Subroutine Description	Page Number
3.5.23	FACTØR	FACTØR	3.5-74
3.5.17	FBS	FBS	3.5-64
3.5.15	FINDC	DECØMP	3.5-57
3.5.15	FINWRT	DECØMP	3.5-54
3.4.17	FNAME	FNAME	3.4-25
3.4.69	FNDPLT	FNDPLT	3.4-117
3.4.15	FREAD	FREAD	3.4-23
3.4.6	FWDREC	FWDREC	3.4-8
3.5.15	GENVEC	DECØMP	3.5-54
3.5.19	GFBS	GFBS	3.5-67
3.4.12	GINØ	GINØ	3.4-15
3.4.61	GINØIØ	GINØIØ	3.4-103
3.4.32	GMMATD	GMMATD	3.4-49
3.4.33	GMMATS	GMMATS	3.4-52
3.3.4	GNFIAT	GNFIAT	3.3-5
3.3.9	GNFIST	GNFIST	3.3-12
3.4.14	GØPEN	GØPEN	3.4-22
3.4.44	IDPLØT	IDPLØT	3.4-75
3.4.45	INTGPT	INTGPX	3.4-76
3.4.45	INTGPX	INTGPX	3.4-76
3.4.46	INTLST	INTLST	3.4-77
3.5.3	INTPK	INTPK	3.5-7
3.5.3	INTPKI	INTPK	3.5-7
3.4.34	INVERD	INVERD	3.4-53
3.4.35	INVERS	INVERS	3.4-54
3.4.47	LINE	LINE	3.4-78
3.4.48	LINE1	LINEi	3.4-79
3.4.48	LINE2	LINEi	3.4-79
3.4.48	LINE3	LINEi	3.4-79
3.4.48	LINE4	LINEi	3.4-79
3.4.48	LINE9	LINEi	3.4-79
3.4.48	LINE10	LINEi	3.4-79

Section Number	Entry Point	Subroutine Description	Page Number
3.4.30	LØCATE	PRELØC	3.4-44
3.5.14	LØØP	SDCØMP	3.5-42
3.4.1	LSHIFT	MAPFNS	3.4-1
3.4.36	MAT	PREMAT	3.4-55
3.4.28	MATDUM	MATDUM	3.4-42
3.5.6	MERGE	PARTN - MERGE	3.5-13
3.4.25	MESAGE	MESAGE	3.4-39
3.5.9	MPART	UPART	3.5-18
3.5.12	MPYAD	MPYAD	3.5-22
3.4.26	MSGWRT	MSGWRT	3.4-40
3.5.15	ØNETWØ	DECØMP	3.5-54
3.4.2	ØPEN	ØPEN	3.4-3
3.4.13	ØPNCØR	ØPNCØR	3.4-20
3.4.1	ØRF	MAPFNS	3.4-1
3.5.2	PACK	PACK	3.5-5
3.4.24	PAGE	PAGE	3.4-38
3.4.24	PAGE1	PAGE	3.4-38
3.4.24	PAGE2	PAGE	3.4-38
3.5.6	PARTN	PARTN - MERGE	3.5-13
3.4.22	PEXIT	PEXIT	3.4-36
3.4.63	PLAMAT	PLAMAT	3.4-106
3.4.67	PLTSET	PLTSET	3.4-113
3.4.30	PRELØC	PRELØC	3.4-44
3.4.36	PREMAT	PREMAT	3.4-55
3.4.39	PRETAB	PRETAB	3.4-67
3.4.37	PRETRD	PRETRD	3.4-64
3.4.38	PRETRS	PRETRS	3.4-66
3.4.49	PRINT	PRINT	3.4-81
3.4.20	RCARD	RCARD	3.4-32
3.5.15	RCØRE	DECØMP	3.5-58
3.4.13	RDCØR	ØPNCØR	3.4-20
3.4.50	RDMØDE	RDMØDX	3.4-83

Section Number	Entry Point	Subroutine Description	Page Number
3.4.50	RDMØDX	RDMØDX	3.4-83
3.4.50	RDMØDY	RDMØDX	3.4-83
3.4.16	RDTRL	WRTTRL	3.4-24
3.4.50	RDWØRD	RDMØDX	3.4-83
3.4.5	READ	READ	3.4-6
3.4.8	REWIND	REWIND	3.4-10
3.4.1	RSHIFT	MAPFNS	3.4-1
3.5.6	RULER	PARTN - MERGE	3.5-14
3.4.51	SCLØSE	SGINØ	3.4-85
3.5.14	SDCØMP	SDCØMP	3.5-30
3.5.8	SDR1B	SDR1B	3.5-17
3.3.6	SEARCH	SEARCH	3.3-8
3.4.43	SELCAM	SELCAM	3.4-74
3.3.3	SEMINT	SEMINT	3.3-3
3.4.51	SEØF	SGINØ	3.4-85
3.4.42	SKPFRM	SKPFRM	3.4-73
3.4.10	SKPFIL	SKPFIL	3.4-12
3.5.20	SØLVER	SØLVER	3.5-69
3.4.51	SØPEN	SGINØ	3.4-85
3.4.31	SØRT	SØRT	3.4-46
3.5.7	SSG2A	SSG2A	3.5-16
3.5.13	SSG2B	SSG2B	3.5-29
3.5.11	SSG2C	SSG2C	3.5-20
3.5.18	SSG3A	SSG3A	3.5-66
3.3.11	SSWTCH	SSWTCH	3.3-15
3.4.52	STPLØT	STPLØT	3.4-87
3.4.51	SWRITE	SGINØ	3.4-85
3.4.53	SYMBØL	SYMBØL	3.4-88
3.5.15	T	DECØMP	3.5-58
3.4.39	TAB	PRETAB	3.4-67
3.4.29	TABPRT	TABPRT	3.4-43
3.4.21	TAPBIT	TAPBIT	3.4-35

Section Number	Entry Point	Subroutine Description	Page Number
3.5.15	TFIN	DECØMP	3.5-58
3.4.54	TIPE	TIPE	3.4-90
3.4.23	TMTØGØ	TMTØGØ	3.4-37
3.5.24	TRANP1	TRANP1	3.5-75
3.4.37	TRANSD	PRETRD	3.4-64
3.5.15	TRANSP	DECØMP	3.5-53
3.4.38	TRANSS	PRETRS	3.4-66
3.5.25	TRNSP	TRNSP	3.5-76
3.3.13	TTLPPG	TTLPPG	3.3-17
3.4.55	TYPE1	TYPEi	3.4-92
3.4.55	TYPE2	TYPEi	3.4-92
3.4.55	TYPE3	TYPEi	3.4-92
3.4.55	TYPE9	TYPEi	3.4-92
3.4.55	TYPE10	TYPEi	3.4-92
3.4.56	TYPFLT	TYPFLT	3.4-94
3.4.57	TYPINT	TYPINT	3.4-96
3.5.4	UNPACK	UNPACK	3.5-10
3.5.9	UPART	UPART	3.5-18
3.4.27	USRMSG	USRMSG	3.4-41
3.4.58	WPLT1	WPLT1	3.4-98
3.4.59	WPLT2	WPLT2	3.4-100
3.4.60	WPLT3	WPLT3	3.4-102
3.4.64	WPLT4	WPLT4	3.4-108
3.4.65	WPLT9	WPLT9	3.4-110
3.4.66	WPLT10	WPLT10	3.4-111
3.4.3	WRITE	WRITE	3.4-4
3.4.13	WRTCØR	ØPNCØR	3.4-20
3.4.16	WRTTRL	WRTTRL	3.4-24
3.3.10	XEØT	XEØT	3.3-14
3.4.11	XGINØ	XGINØ	3.4-13

Section Number	Entry Point	Subroutine Description	Page Number
3.5.15	XLØØP	DECØMP	3.5-60
3.4.1	XØRF	MAPFNS	3.4-1
3.4.19	XRCARD	XRCARD	3.4-27
3.3.8	XSEMXX	XSEMXX	3.3-11
3.3.1	XSEM1	XSEM1	3.3-1
3.3.7	XSEM2	XSEMi	3.3-9
3.3.7	XSEM3	XSEMi	3.3-9
3.3.7	XSEM4	XSEMi	3.3-9
3.3.7	XSEM5	XSEMi	3.3-9
3.3.7	XSEM6	XSEMi	3.3-9
3.3.7	XSEM7	XSEMi	3.3-9
3.3.7	XSEM9	XSEMi	3.3-9
3.3.7	XSEM10	XSEMi	3.3-9
3.3.7	XSEM11	XSEMi	3.3-9
3.3.7	XSEM12	XSEMi	3.3-9
3.3.7	XSEM13	XSEMi	3.3-9
3.3.7	XSEM14	XSEMi	3.3-9
3.5.1	ZBLPKI	BLDPK	3.5-1
5.5.3	ZNTPKI	INTPK	3.5-7

3.3 EXECUTIVE SUBROUTINE DESCRIPTIONS.

3.3.1 XSEM1 (Executive Sequence Monitor, Preface).

3.3.1.1 Entry Point: XSEM1.

3.3.1.2 Purpose

To initiate the execution of the NASTRAN Preface.

3.3.1.3 Calling Sequence

CALL XSEM1

3.3.1.4 Method

Subroutine BTSTRP is called to initialize machine dependent data, and then subroutine SEMINT is called to execute the program Preface (i.e. input file processors and DMAP program compiler). After initiating the problem, modules are called as directed by the ØSCAR until a module is encountered in the ØSCAR that does not reside in link 1 at which time XSEM1 calls subroutine ENDSYS to load a new link.

3.3.1.5 Design Requirements

XSEM1 must reside in the core resident portion of link 1. Link 1 is not re-entrant which means that once the program leaves link 1 it can never transfer control back to link 1. Functional DMAP modules can not reside in link 1. Open core is used for a GINØ buffer with named common block /ESFA/ defining the beginning of open core. See the second paragraph of the design requirements section of the subroutine description XSEMi (see section 3.3.7) for details on files, data blocks, and common blocks necessary for operation.

3.3.2 BTSTRP (Bootstrap Generator).

3.3.2.1 Entry Point: BTSTRP.

3.3.2.2 Purpose

Determines the machine type and initializes the machine dependent constants and masks within the NASTRAN system block data program (SEMDBD).

3.3.2.3 Calling Sequence

CALL BTSTRP

3.3.2.4 Method

The machine type (IBM 7094, IBM S/360, Univac 1108, CDC 6600) is determined by inspection of the machine binary word length and the known methods of representing negative integers (sign and magnitude or ones/twos complement) using the following algorithm:

1. If the ones complement (COMPLF see section 3.4.1) of -1 is greater than 2, the machine is the IBM 7094. If not, the machine is an IBM S/360, Univac 1108 or CDC 6600. (i.e., only the sign and magnitude representation of -1 on the 7094 will yield a large (> 2) positive value when complemented.)

2. Shift (RSHIFT see section 3.4.1) a binary machine word of all 1's to the right thirty-two binary places. Compare the resulting value to 15. If the value is less than fifteen, the machine is the 32 bit IBM S/360; equal to fifteen, the 36 bit Univac 1108; and greater than fifteen, the 60 bit CDC 6600.

Once the machine type is known, the proper constants and masks are selected from assembled tables.

3.3.2.5 Design Requirements

This subroutine must be modified if it is to operate with other than the four machine types listed above.

3.3.3 SEMINT (Sequence Monitor Initialization).

3.3.3.1 Entry Point: SEMINT.

3.3.3.2 Purpose

To execute the Preface of a NASTRAN problem solution.

3.3.3.3 Calling Sequence

CALL SEMINT

3.3.3.4 Method

The first card of the NASTRAN data deck is read from the system input file and its image stored in blank CØMMØN. XRCARD is called to convert the card. If the name of the card is NASTRAN, the card is echoed and keywords are identified and appropriate words of /SYSTEM/ are reset to the input values. If an unidentified keyword is detected, or the card has a format error, a message is printed and the NØGØ flag is turned on. The first word of blank CØMMØN is set to one if the card was a NASTRAN card, to zero otherwise. Then GNFIAT is called to generate the initial FIAT. XCSA is called to read and process the Executive Control Deck. IFP1 is called to read and process the Case Control Deck. XSØRT is called to read and sort the Bulk Data Deck. If bulk data is present, IFP is called to process it. If the problem is a conical shell problem, IFP3 is called to further process the bulk data. If the current run is to prepare a User's Master File, UMFEDT is called and control is returned to XSØRT for each new problem to be written on the UMF. Otherwise, XGPI is called to perform General Problem Initialization and then return is made to XSEM1 signifying completion of the Preface.

3.3.3.5 Design Requirements

If the NASTRAN card is present, it must be the first card of the data deck.

3.3.3.6 Diagnostic Messages

UNIDENTIFIED NASTRAN KEYWØRD _____. ACCEPTABLE KEYWØRDS FØLLØW--

BUFFSIZE

CØNFIG

MAXFILES

MAXØPEN

SYSTEM

Self-explanatory.

NASTRAN CARD DØES NØT HAVE CØRRECT FØRMAT.

Typical errors include non-integer values or continuation of the card following an = sign.

See section 6.3.1 for further details on the NASTRAN card.

3.3.4 GNFIAT (Generate FIAT).

3.3.4.1 Entry Point: GNFIAT.

3.3.4.2 Purpose

Determines the number of logical files available within the computer hardware and software configuration and places an entry for each into FIAT or XFIAT.

3.3.4.3 Calling Sequence

CALL GNFIAT

GNFIAT must be called once and only once as the first call from the preface.

3.3.4.4 Method

Each computer configuration has its own independent subroutine to accomplish the necessary functions of GNFIAT. The subroutine interrogates unit blocks, data definition cards, file tables, etc. to determine the number of logical files available within the configuration. As each logical file is sensed, it is determined whether the file has been assigned to a physical magnetic tape or some bulk storage device such as disk or drum. Each file has a logical name and/or number for identification. These file ID's are stored in FIAT, XFIAT or both depending on several factors. As the file ID is stored, a physical tape flag is set where appropriate. The factors that determine FIAT vs. XFIAT storage are as follows: 1) the first PFIST (see section 2.4 for a description of the FIST) files sensed are always entered into XFIAT, 2) except for the first file (always the PØØL), all of the first PFIST files without tape flags are also entered into FIAT, and 3) all other files are entered into FIAT only.

3.3.4.5 Design Requirements

Since GNFIAT routines are computer hardware/software dependent, operational requirements may differ at various times. See appropriate commented assembly listing if difficulties or error codes are encountered.

3.3.5 ENDSYS (End-of-Link).

3.3.5.1 Entry Points: ENDSYS, BGNSYS.

3.3.5.2 Purpose

For ENDSYS, to save various NASTRAN core-resident Executive Tables on a scratch file for use in communicating with the next link requested.

For BGNSYS, to restore the NASTRAN Executive Tables saved by ENDSYS and to position the ØSCAR at the correct entry to be executed in the resident link.

3.3.5.3 Calling Sequences

CALL ENDSYS(LINK,X,REWFLG)

LINK - BCD name of the link. The naming convention is: NSO1 = link 1, NSO2 = link 2, etc.

X - Dependent on machine type. For the IBM 7094 only, X (6 BCD characters) specifies the unit where the links are stored. Not used on other machines.

REWFLG = 0 indicates LINK is ahead of current link (i.e. we are going from link N to link N + K, K > 0). IBM 7094 only; not used on other machines.

REWFLG = 1 indicates LINK is behind current link (i.e. we are going from link N to link N + K, K < 0). IBM 7094 only; not used on other machines.

CALL BGNSYS.

3.3.5.4 Method

For ENDSYS, a search is made for an empty file and when found the Executive Tables are written (saved) on it. A pointer to the save file is saved in blank common or written on a system file for use by BGNSYS when the new link is loaded. Subroutine SEARCH is then called to load the requested link.

BGNSYS is called after a new link is loaded. The pointer to the save file containing the Executive Tables is obtained from either blank common or a system file, and the Executive Tables are reloaded into core. The ØSCAR is positioned at the correct entry to be executed, and a RETURN is made to the calling routine.

3.3.5.5 Design Requirements

Program links are usually considered to be physically separate programs, essentially independent of one another except for the fact that the operating system executive (not the NASTRAN executive) provides a means by which control can be transferred from one link to another when requested by the user. The means by which the operating system executive transfers control from one link to another is dependent upon the machine and the system being used. For some future machines there may be no means for building physically separate links so the links become logical subsets of one huge program.

No matter how the links are formed it is necessary, when transferring from one link to another, that all file assignments be preserved as well as their status (i.e. don't rewind the tapes).

Open core is used for GINØ buffer area and the beginning of open core is defined by named common block /ESFA/.

If no save file is available or if any of the Executive Tables to be written exceeds 900 words, the job is terminated.

3.3.6 SEARCH (Search, Load, and Execute Link).

3.3.6.1 Entry Point: SEARCH.

3.3.6.2 Purpose

SEARCH locates (searches for) a particular link of the NASTRAN system on the Link Storage File, loads the link into the computer memory and transfers execution control to the link entry point XSEMi, i = 2,3,....

3.3.6.3 Calling Sequence

```
CALL SEARCH(LKNAM,LKFIL,REW)
```

LKNAM = 4 character symbolic name of link, i.e., NSO1, NSO2 for link 1, link 2, etc.

LKFIL = symbolic name of the Link Storage File (IBM 7094 only)

REW = set non-zero to position a sequential Link Storage File to its beginning (IBM 7094 only)

3.3.6.4 Method

SEARCH is machine dependent. It interfaces with the machine operating system to provide a multi-link capability. Each link is a somewhat arbitrary part of the complete NASTRAN system. The division into links was necessary only because of the size limitation for program complexes imposed by the various operating systems. The linking technique for each machine is discussed in section 5 of the Programmer's Manual.

3.3.6.5 Design Requirements

Only the IBM 7094 system requires the Link Storage File to be named (LKFIL) and, since it is sequential, provides the capability of rewinding it following a SEARCH call. All other systems provide random access (disk, drum) Link Storage Files.

3.3.6.6 Diagnostic Messages

Individual SEARCH subroutines may abnormally terminate due to hardware malfunction. See appropriate commented assembly listing if difficulties or error codes are encountered.

3.3.7 XSEMi (Link i Main Program, i = 2,3, ...

3.3.7.1 Entry Point: XSEMi.

3.3.7.2 Purpose

To get the next module to be executed from the ØSCAR, initialize the module and call it if it is in link i, or transfer to the link in which module resides if it is not in link i.

3.3.7.3 Calling Sequence

Example: CALL XSEM2, where XSEM2 is the entry point of link 2

3.3.7.4 Method

Subroutine BTSTRP is called to initialize machine dependent data, and then subroutine BGNSYS is called to reload Executive Tables saved from the previous link.

The next ØSCAR entry is read into core and processed. If the entry is for a functional module, subroutine GNFIST is called to link files with input, output and scratch data blocks needed by the module. Variable parameter values needed by the module are transferred to blank common from table VPS which resides in named common block /XVPS/. Constant values in the ØSCAR entry parameter section are transferred to blank common.

The link specification table in named common block /XLINK/ is examined to see if the module resides in this link. If it does, the module is called. Upon returning from the module, the diagnostic message queue is checked and message writer MSGWRT is called if there are messages queued. Begin and end execution times are printed out for functional modules.

The next ØSCAR entry is read and the process is repeated until a module is encountered which does not reside in this link, at which time subroutine ENDSYS is called to initiate loading of the link containing the module.

3.3.7.5 Design Requirements

XSEMi must reside in the core resident portion of link i. Link i is re-entrant which means program control can be transferred to this link as often as needed. Open core is used for a GINØ buffer with named common block /ESFA/ defining the beginning of open core. An ØSCAR entry cannot be greater than 200 words.

Files, data blocks and named common blocks needed by XSEMi are listed below, along with type of access required (i.e. fetch and/or store data) and reasons for use.

1. Data Pool File - fetch. Contains XØSCAR data block.
2. XØSCAR - fetch. Contains ØSCAR entry to be processed.
3. Common /XLINK/ - fetch. Contains link specification table.
4. Common /XFIST/ - store. Initialized prior to calling GNFIST.
5. Common /XPFIST/ - fetch. Contains parameter needed to initialize FIST table.
6. Common /ØSCENT/ - fetch. Contains ØSCAR entry to be processed.
7. Common /ESFA/ - store. Defines beginning of open core area used by GINØ buffer.
8. Common /XVPS/ - fetch. Contains variable parameter values needed to initialize module to be executed.
9. Common /MSGX/ - fetch. Contains diagnostic message queue.
10. Common /SEM/ - fetch. Contains BCD names of links NS01, NS02, ...

3.3.7.6 Diagnostic Messages

A message is written if the module to be executed required more files than are available. The job is then terminated.

3.3.8 XSEMXX (Sequence Monitor - Deck Generator).

3.3.8.1 Entry Point: XSEMXX.

3.3.8.2 Purpose

To provide a model from which all other XSEMi (i = link number) subroutines except XSEM1 can be made.

See section 6.11, which discusses how to generate a link driver subroutine.

3.3.9 GNFIST (Generate FIST)

3.3.9.1 Entry Point: GNFIST.

3.3.9.2 Purpose

To set up the proper linkage between data blocks and the files they reside on in preparation for executing the functional module requiring the data blocks.

3.3.9.3 Calling Sequence

CALL GNFIST(DBN,FISTNM,MØDNØ)

DBN - Data block name (Two word BCD array - 8 characters total)

FISTNM - Data block identification (GINØ file number) used by functional module (integer).

MØDNØ - ØSCAR record number of functional module to be executed (integer). MØDNØ indicates to the calling routine what action was taken by GNFIST.

MØDNØ ≠ 0, data block assigned a file or it was purged.

MØDNØ = 0, fatal error detected.

MØDNØ < 0, data block not assigned a file, GNFIST called Executive Segment File Allocator (XSFA)

3.3.9.4 Method

If the data block is purged, GNFIST returns to the calling routine with MØDNØ ≠ 0. A data block is purged if it is an input which has not been generated or its status is purged or DBN = 0.

If an input data block resides on the Data Pool File and needs to be unpooled, GNFIST calls the file allocator (XSFA) to unpool all inputs to the module which reside on the Data Pool File that need to be unpooled. GNFIST then returns to the calling routine with MØDNØ < 0. The other condition under which XSFA is called is if a file has <u>not</u> been allocated to a non-purged output data block or scratch data block needed by the module.

A file is allocated to a data block when the data block name appears in the FIAT table, located in named common block /XFIAT/, as unpurged. Input, output and scratch data blocks which have been assigned to a file and are required by the functional module, have their FISTNM's entered

in the FIST table which is located in named common block /XFIST/. FIST entries are linked to the DBN's in the FIAT table which in turn links the data block to a file. This completes the linking of functional module data blocks to their files.

Output data blocks cannot reside on the Data Pool File, so GNFIST checks for this and if found, the DBN and all DBN's equivalenced to it are deleted from the DPL table located in named common /XDPL/.

3.3.9.5 Design Requirements

GNFIST resides in the core resident portion of a link. It does not use open core and the only restriction is that the FIST table be large enough to hold all FISTNM's for a module.

The named common blocks needed by GNFIST are listed below, along with type of access required (i.e. fetch and/or store data) and reasons for use.

1. CØMMØN/XFIST/ - Store.

   Used to store FISTNM's and link FISTNM's with their corresponding DBN's in FIAT.

2. CØMMØN/XFIAT/ - Fetch and store.

   Used to determine status of DBN's. The FIAT table is altered if unpooling of input data blocks is necessary.

3. CØMMØN/XDPL/ - Fetch and store.

   Used to determine status of input DBN's and is altered if output DBN's appear in it.

4. CØMMØN/ØSCENT/ - Fetch.

   Contains ØSCAR entry for functional module to be executed. Used to alter FIAT when input DBN's need to be unpooled.

3.3.9.6 Diagnostic Messages

GNFIST detects overflow in FIST table and sends message to terminate job.

3.3.10 XEØT (End-of-Tape).

3.3.10.1 Entry Point: XEØT.

3.3.10.2 Purpose

To prepare and send to the computer operator, messages instructing him what to do when end-of-tape has been encountered on the Old Problem Tape (ØPTP) or the New Problem Tape (NPTP).

3.3.10.3 Calling Sequence

CALL XEØT(ID,ØREEL,NREEL,BUF)

ID - BCD name NPTP or ØPTP

ØREEL - Number of reel to be dismounted - integer.

NREEL - Number of new reel to be mounted - integer.

BUF - GINØ buffer used by NPTP or ØPTP.

3.3.10.4 Method

A check is made to see if tape has multi-reel capability. If not, a fatal message is issued and job is terminated. The operator messages are generated and issued and the old reel is re-wound and unloaded. A check is made to see if correct new reel has been mounted and then a return is made to calling program.

3.3.10.5 Design Requirements

XEØT must be accessible to routines XGPI and XCHK.

3.3.10.6 Diagnostic Messages

A message is issued if tape does not have multi-reel capability.

3.3.11 SSWTCH (Sense Switches).

3.3.11.1 Entry Point: SSWTCH.

3.3.11.2 Purpose

To indicate to the calling routine whether or not a specified sense switch is set.

3.3.11.3 Calling Sequence

CALL SSWTCH(SS,F)

SS - Sense switch number - integer. $1 \leq SS \leq 31$.

F - Flag indicating status of SS

F = 0 if SS not on

F = 1 if SS is on

3.3.11.4 Method

Named common block /SYSTEM/ contains the word which contains the sense switch settings. Bit 1 of the word corresponds to sense switch 1, bit 2 corresponds to sense switch 2, etc. If the bit corresponding to SS is on then F = 1, if not then F = 0.

Note that sense switches are set by the user via the DIAG card in the Executive Control Deck and not through physical sense switches set by the computer operator.

3.3.11.5 Design Requirements

SSWTCH resides in the core resident portion of a link.

3.3.12 CØNMSG (Console Message Writer).

3.3.12.1 Entry Point: CØNMSG

3.3.12.2 Purpose

Writes the current time and a NASTRAN system message onto the system output device and (if the computer configuration permits) onto the on-line operator's console device.

3.3.12.3 Calling Sequence

```
CALL CØNMSG(MSG,CNT,YN)
```

MSG - Array name containing message.

CNT - Number of 4-character words in message (integer).

YN - 1 = yes, 0 = no. Print on-line device if yes and available.

3.3.12.4 Method

A computer real-time and/or job clock is interrogated. The number of message words indicated is sent to the system output device (usually printer) along with the clock reading(s). If the computer configuration permits and the yes/no switch is set yes, the same clock reading(s) and message is sent to the operator's console device (usually typewriter).

3.3.12.5 Design Requirements

Only the left-most four characters from each computer word are extracted and sent to the output device(s).

3.3.13 TTLPPG (Title Page Writer).

3.3.13.1 Entry Point: TTLPPG.

3.3.13.2 Purpose

To print on the system output file 2 title pages identifying the job as a NASTRAN problem, identifying the current NASTRAN source and object libraries and identifying the current Rigid Format series.

3.3.13.3 Calling Sequence

CALL TTLPPG.

3.3.13.4 Method

TTLPPG is called as the first executable statement in the Preface driver SEMINT.

3.3.13.5 Design Requirements

Whenever a new version of NASTRAN is installed, a level number of the form N.M must be updated in TTLPPG, where N is the source library level number and M is the object library version number. Whenever N is incremented by 1, M is reset to zero. Minor changes in the system are accomplished by updating the necessary routines, thereby generating new object (relocatable) decks for these routines. M is incremented by 1 but the source library and hence N remain unchanged. The Rigid Format series is identified by a letter. A Rigid Format series can change only when N is changed.

3.4 UTILITY SUBROUTINE DESCRIPTIONS.

3.4.1 MAPFNS (Machine Word Functions).

3.4.1.1 Entry Points: LSHIFT, RSHIFT, ANDF, ØRF, XØRF, CØMPLF, CØRSZ, CØRWDS.

3.4.1.2 Purpose

To perform basic computer word manipulations by standard binary digit (bit) operations. The manipulations are performed over the complete memory word length for the particular hardware. Also, to determine the size of open core (CØRSZ) and the absolute difference between locations in core (CØRWDS).

3.4.1.3 Calling Sequence

All machine word functions are executed as FØRTRAN integer function subroutines with integer arguments.

3.4.1.4 Method

The method employed within each function will be described following the separate function examples.

3.4.1.5 Entries

K = LSHIFT (I,N)

The entire bit structure of word I is shifted left N places and the resulting word replaces word K. Word I is unchanged. High-order bits shifted out are lost. Zeros are supplied to vacated low-order positions. The shift is logical; no special provision is made for the sign position.

K = RSHIFT (I,N)

The entire bit structure of word I is shifted right N places and the resulting word replaces word K. Word I is unchanged. Low-order bits shifted out are lost. Zeros are supplied to vacated high-order positions. The shift is logical; no special provision is made for the sign position.

K = ANDF (I,J)

A logical product of the bits within word I and word J is formed and stored into word K. Words I and J are unchanged.

K = ØRF (I,J)

A logical sum of the bits within word I and word J is formed and stored into word K. Words I and J are unchanged.

K = XØRF (I,J)

The modulo-two sum (exclusive or) of the bits within word I and word J is formed and stored into word K. Words I and J are unchanged.

K = CØMPLF (I)

The ones complement of the bits within word I is formed and stored into word K. Word I is unchanged.

K = CØRSZ (I,J)

The size of open core is computed and stored in location K through this function. Location I is normally the address of a labeled common cell defining the beginning of a particular open core area. Location J is normally the address of blank common (usually thought to be the end of a particular open core area). On computer memory configurations where blank common does not define the end of open core, CØRSZ ignores location J and substitutes a correct end value. The arguments I and J may be interchanged without affecting results.

K = CØRWDS (I,J)

The absolute difference plus 1 between the addresses of locations I and J is computed and stored into word K. Words I and J are unchanged.

3.4.1.6 Design Requirements

MAPFNS is written in assembly language.

3.4.2 ØPEN (Initiate Activity on a File).

3.4.2.1 Entry Point: ØPEN.

3.4.2.2 Purpose

To initiate activity on the requested file.

3.4.2.3 Calling Sequence

CALL ØPEN($n,NAME,BUFF,ØP)

n - FØRTRAN statement number defining the return to be taken in the event NAME is not in the FIST (i.e. the data block is purged).

NAME - GINØ file name of the data block which is to be read or written(see section 1.6.4.1).

BUFF - An array whose dimension equals the contents of the first word of /SYSTEM/ which will be used by GINØ while the file is open.

ØP = { 0, open file to read with rewind
1, open file to write with rewind
2, open file to read without rewind
3, open file to write without rewind

3.4.2.4 Method

ØPEN stores parameters in /GINØX/ and then calls XGINØ which searches the FIST for a name match and picks up from the FIAT the unit to which the data block is assigned. The position of the buffer is determined relative to /XNSTRN/.

This index is saved in the BUFADD array in /GINØX/. BUFADD is searched to determine if any other buffer overlaps the buffer currently assigned. GINØ is called to initiate activity for the file.

3.4.2.5 Design Requirements

The address of the buffer assigned must be greater than the address of /XNSTRN/.

3.4.2.6 Diagnostic Messages

The following system fatal errors may be issued by ØPEN:

3006

3012

3040

3.4.3 WRITE (Write Data in a Logical Record).

3.4.3.1 Entry Point: WRITE.

3.4.3.2 Purpose

To write a logical record, or portion of a logical record, on the requested file.

3.4.3.3 Calling Sequence

```
CALL WRITE(NAME,BLØCK,N,EØR)
```

NAME - GINØ file name of the data block which is to be written (see section 1.6.4.1).

BLØCK- An array of dimension ≥ N containing the data words to be written.

N - The number of words to be written - integer - input.

EØR = { 0, the N words to be written by this call do not end the logical record, i.e. subsequent WRITE calls will provide additional data to be written in the current logical record.
{ 1, the N words to be written by this call end the logical record.

3.4.3.4 Method

WRITE stores parameters in /GINØX/ and then calls XGINØ which in turn calls GINØ to perform the actual processing of the call.

3.4.4 CLØSE (Terminate Activity on a File).

3.4.4.1 Entry Point: CLØSE.

3.4.4.2 Purpose

To terminate activity on the requested file.

3.4.4.3 Calling Sequence

```
CALL CLØSE(NAME,ØP)
```

NAME - GINØ file name of the data block to be closed (see section 1.6.4.1).

ØP
- 1, if file was opened to write, write end-of-file and rewind.
  If file was opened to read, rewind only.
- 2, close file at current file position (no end-of-file, no rewind).
- 3, if file was opened to write, write end-of-file and position file in front of end-of-file mark.
  If file was opened to read, same as ØP = 2.

If the requested file is not in the FIST or is not currently open, a normal return is given and no operation takes place.

If the file was opened for output and the last logical record has not been written, it is written prior to honoring the ØP request.

The buffer assigned when the file was opened is released and is available to the user on return.

3.4.4.4 Method

CLØSE stores parameters in /GINØX/ and then calls XGINØ. If XGINØ returns indicating the file is not in the FIST or not open, an immediate return is given to the user. Otherwise XGINØ calls GINØ to process the call.

3.4.5 READ (Read Data From a Logical Record).

3.4.5.1 Entry Point: READ.

3.4.5.2 Purpose

To read the contents of a logical record, or portion of a logical record, from the requested file.

3.4.5.3 Calling Sequence

CALL READ($$n_1$$,$$n_2$$,NAME,BLØCK,N,EØR,M)

$$n_1$$ - FØRTRAN statement number defining the return to be taken in the event an end-of-file is encountered by this READ operation.

$$n_2$$ - FØRTRAN statement number defining the return to be taken at the end of the READ operation whenever the end-of-logical-record is encountered prior to transmitting the requested number of data words.

NAME - GINØ file name of the data block which is to be read (see section 1.6.4.1).

BLØCK - An array of dimension ≥ N, where the words read will be stored.

N - { N > 0: The number of words to be read and stored at BLØCK - integer - input.
N ≤ 0: The number of words to be skipped, i.e., read but not stored at BLØCK.
Integer - input. }

EØR - { 0, subsequent calls to READ for the current logical record are expected.
1, the current call is the last call for the current logical record. The file will be positioned to the beginning of the next logical record before returning. }

M - If return to $$n_2$$ is given, the number of words actually read is stored in M. In no other case are the contents of M changed.

Whenever return to $$n_2$$ is given, the file is positioned to the beginning of the next logical record regardless of the setting of EØR.

A return to $$n_1$$ is possible only when a call to READ is given when the file is positioned at the beginning of a logical record.

3.4.5.4 Method

READ stores parameters in /GINØX/ and then calls XGINØ which in turn calls GINØ to perform the actual processing of the call.

3.4.6 FWDREC (Forward Space One Logical Record).

3.4.6.1 Entry Point: FWDREC.

3.4.6.2 Purpose

To position the requested file forward one logical record.

3.4.6.3 Calling Sequence

CALL FWDREC($n,NAME)

n - FØRTRAN statement number defining the return to be taken in the event an end-of-file is encountered.

NAME - GINØ file name of data block to be positioned forward (see section 1.6.4.1).

This call will always position the file to the beginning of the next logical record unless an end-of-file is encountered.

3.4.6.4 Method

FWDREC stores parameters in /GINØX/ and then calls XGINØ which in turn calls GINØ to process the call.

3.4.7 BCKREC (Backspace One Logical Record).

3.4.7.1 Entry Point: BCKREC.

3.4.7.2 Purpose

To position the requested file backward one logical record.

3.4.7.3 Calling Sequence

CALL BCKREC (NAME)

NAME - GINØ file name of data block to be positioned backward (see section 1.6.4.1).

If the file is positioned in the middle of a logical record, the file is repositioned to the beginning of that record. Otherwise, the file is positioned to the beginning of the previous logical record.

If the file is positioned at the beginning of file, no operation occurs and a normal return is given.

3.4.7.4 Method

BCKREC stores parameters in /GINØX/ and then calls XGINØ which in turn calls GINØ to process the call.

3.4.8 REWIND (Position File to the Load Point).

3.4.8.1 Entry Point: REWIND

3.4.8.2 Purpose

To rewind the requested file.

3.4.8.3 Calling Sequence

CALL REWIND(NAME)

NAME - GINØ file name of the data block to be rewound (see section 1.6.4.1).

Rewind given for an output file has the effect of erasing any data which has been written on the file.

3.4.8.4 Method

REWIND stores parameters in /GINØX/ and then calls XGINØ which in turn calls GINØ to process the call.

3.4.9 EØF (Write an End-of-File).

3.4.9.1 Entry Point: EØF.

3.4.9.2 Purpose

To write an end-of-file on the requested file.

3.4.9.3 Calling Sequence

CALL EØF(NAME)

NAME - GINØ file name of data block on which end-of-file is to be written(see section 1.6.4.1.)

The file must be open to write at the time of this call.

3.4.9.4 Method

EØF stores parameters in /GINØX/ and then calls XGINØ which in turn calls GINØ to process the call.

3.4.10 SKPFIL (Skip Files Forward or Backward).

3.4.10.1 Entry Point: SKPFIL.

3.4.10.2 Purpose

To position the requested file forward or backward a stated number of files.

3.4.10.3 Calling Sequence

CALL SKPFIL(NAME,N)

NAME - GINØ file name of the data block to be repositioned (see section 1.6.4.1).

N - The number of files to be skipped. N > 0 means forward skip, N < 0 means backward skip, N = 0 means no operation - integer - input.

Notes:

1. Following a forward skip, the file is positioned at the beginning of the first logical record (i.e. immediately after the end-of-file mark).

2. Following a backward skip, the file is positioned immediately in front of the end-of-file mark (or at the beginning-of-unit).

3. Request to skip backward from the beginning-of-unit is ignored and the file remains positioned at the beginning-of-unit.

4. SKPFIL backward on a file opened to write has the effect of "erasing" file(s) written.

3.4.10.4 Method

SKPFIL stores parameters in /GINØX/ and calls XGINØ which in turn calls GINØ to process the call.

3.4.11 XGINØ (GINØ Utility Routine).

3.4.11.1 Entry Point: XGINØ.

3.4.11.2 Purpose

To convert the GINØ file name to a unit number, retrieve the buffer assigned to the file and call GINØ.

3.4.11.3 Calling Sequence

CALL XGINØ(n_1,n_2,A,M)

CØMMØN/GINØX/LGINØX,FILEX,EØR,ØP,ENTRY,LSTNAM,N,NAME,NTAPE,XYZ(2),UNITAB(75),BUFADD(75)

ENTRY = { 1, call from subroutine ØPEN
2, call from subroutine WRITE
3, call from subroutine READ
4, call from subroutine CLØSE
5, call from subroutine BCKREC
6, call from subroutine FWDREC
7, call from subroutine SKPFIL
8, call from subroutine EØF
9, call from subroutine REWIND } integer - input

NAME - GINØ file name of data block for which activity is requested - integer - input.

FILEX - Unit number to which NAME is assigned - integer - output.

LSTNAM - On entry to XGINØ, GINØ name from previous call. On exit from XGINØ, LSTNAM = NAME - integer - input and output.

NTAPE = { 0, file does not reside on tape
1, file resides on tape } integer - output.

n_1 - { if ENTRY = 1 or 4, FØRTRAN statement number defining return in the event NAME is not in FIST.
if ENTRY = 3 or 6, FØRTRAN statement number defining return in the event an end-of-file is encountered.

n_2 - If ENTRY = 3, FØRTRAN statement number defining end-of-logical-record prior to completion of requested read.

A - If ENTRY = 2 or 3, A is the user block (from/to) which data words are written/read.

M - If ENTRY = 3 and return to n_2 is given, M = number of words read - integer - output.

3.4.11.4 Method

The FIST is searched for a name match. If found, the pointer to FIAT is used to pick up the unit number and tape flag. If not found, a non-standard return is given (ENTRY = 1 or 4) or a fatal message is generated. The address of the buffer assigned to the file is picked up and GINØ is called to execute the requested operation.

3.4.11.5 Design Requirements

XGINØ is designed as an integral part of the GINØ collection of routines for use only by ØPEN, READ, WRITE, etc.

The BUFADD table must be initialized to zero prior to the first call in a NASTRAN execution.

3.4.11.6 Diagnostic Messages

The following system fatal errors may be issued by XGINØ:

3010

3021

3.4.12 GINØ (General Input/Output Routine).

3.4.12.1 Entry Point: GINØ.

3.4.12.2 Purpose

To provide general purpose Input/Output services to higher level routines (READ, WRITE, etc.) in the NASTRAN program.

3.4.12.3 Calling Sequence

CALL GINØ(n_1,n_2,BUFF,A,M)

CØMMØN/GINØX/LGINØX,FILEX,EØR,ØP,ENTRY,LSTNAM,N,NAME,NTAPE,XYZ(2),UNITAB(75),BUFADD(75)

ENTRY = { 1, ØPEN Operation
2, WRITE Operation
3, READ Operation
4, CLØSE Operation
5, BCKREC Operation
6, FWDREC Operation
7, SKPFIL Operation
8, EØF Operation
9, REWIND Operation } Input, integer

FILEX - Unit number of file - integer - input.

EØR - End-of-record flag (see READ, WRITE) - integer - input.

ØP - Operation code (see ØPEN, CLØSE) - integer - input.

N - Number of words to write/read or number of files to skip (see WRITE, READ, SKPFIL) - integer - input.

BUFF - Address of buffer assigned to FILEX.

n_1 - FØRTRAN statement number defining return in the event an end-of-file is encountered (READ and FWDREC operations only).

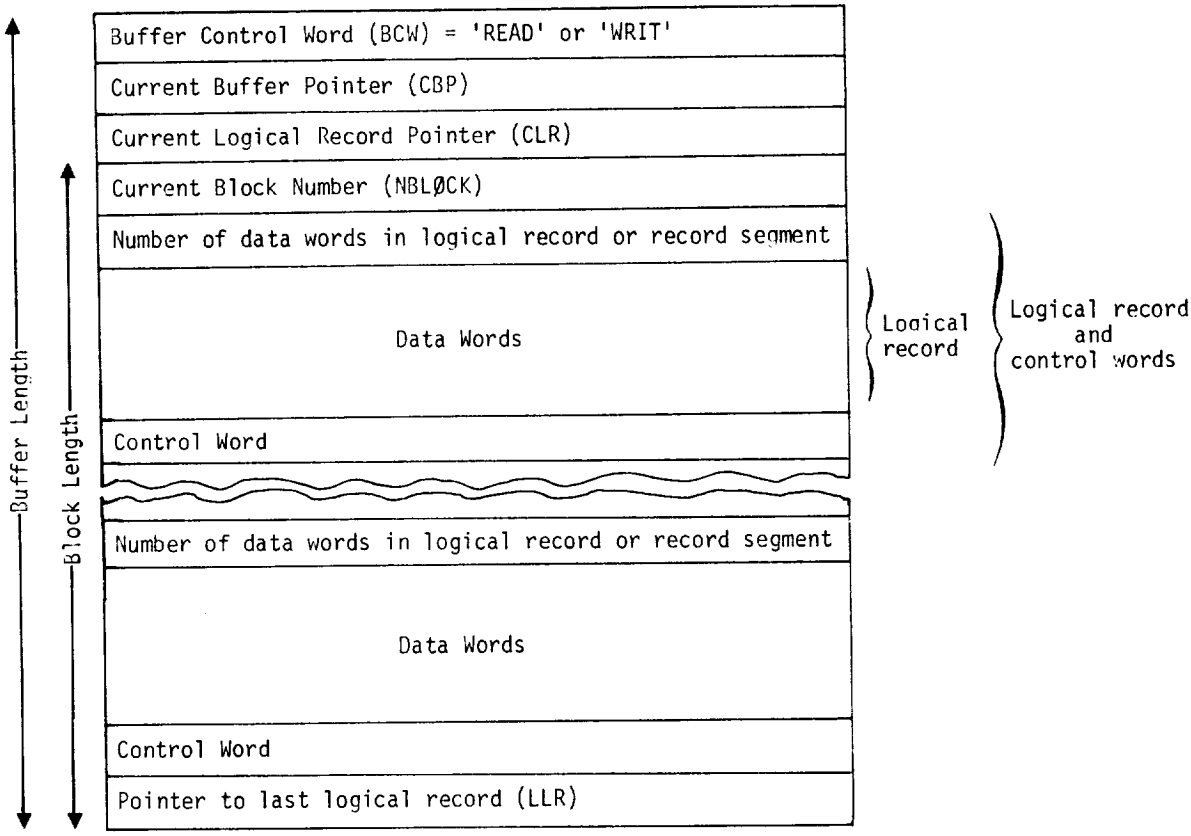
n_2 - FØRTRAN statement number defining return in the event an end-of-record is encountered prior to transmitting the requested number of words (READ operations only).

A - User block (see WRITE, READ).

M - Number of words actually read if end-of-record encountered and return to n_2 is given (see READ) - integer - output.

3.4.12.4 Method

GINØ blocks all logical records into fixed length physical records (blocks) for writing and reading. A description of the GINØ buffer follows:



Control word = 4*K + 2*F2 + F1 where

F1 = 1 indicates continued record

F2 = 0 indicates last segment of continued record

if F1 = 0, K = number of words in logical record

if F2 = 1, K = 65536*(Block No.) + CLR of 1st segment of logical record

Processing of each operation takes place as follows:

1. ØPEN. If a rewind is requested, the file is rewound and the first three words of the buffer are initialized followed by a return. Otherwise, the current file position is retrieved from UNITAB. If the file is logically between blocks, action occurs as above. Otherwise, the next block is read and the pointer to the current logical record is restored. If the block read is not the expected block, a recovery attempt is made.

2. WRITE. For the first call to write in a logical record, NBLØCK and CLR are saved in UNITAB. User data is transferred to the buffer. If the buffer is filled, a control word indicating continuation is placed, the block is written and the logical record is continued in the next block. After data transfer to the buffer, EØR is tested. If on, a control word is placed in the buffer and pointers are moved to the beginning of a new logical record.

3. READ. If the buffer is empty, the next block on the file is read. If positioned at the beginning of a logical record, a test for logical end-of-file is made. If on, return to n_1 is given. Otherwise data is transferred from the buffer to the user area. If the logical end-of-record is encountered before the requested number of words have been transmitted, M is set with the number of words read, and return to n_2 is given. If the logical record is continued to the next block, the next block is read. After the transfer of data is complete, EØR is tested. If on, pointers are moved to the beginning of the next logical record.

4. CLØSE. If the file was opened to read, ØP is tested. If ØP = 1, the file is rewound. If ØP ≠ 1, the file is backspaced one block unless the file is logically between blocks. In either case, the logical position of the file (NBLØCK and CLR) is saved in UNITAB, BCW is set to zero and return is made. If the file was opened to write and CRP ≠ CLR, control words for the last logical record are stored in the buffer. If ØP = 1 or 3, a logical end-of-file is placed. The current block is written on the file. If ØP = 1, the file is rewound, otherwise the file is backspaced one block. The logical position of the file is saved in UNITAB, BCW is set to zero and return is made.

5. BCKREC. If the buffer in core is empty, the file is backspaced one block, the block is read and LLR is used to set the pointers to the last logical record in the block. If the file is logically positioned in the middle of a logical record, pointers are reset to the beginning of a logical record. If the file is positioned at the beginning of a logical record, pointers are reset to the beginning of the previous logical record.

6. FWDREC. N is set to zero and EØR to 1 and the code in the READ portion of GINØ is used to position the file at the beginning of the next logical record.

7. SKPFIL. If a forward skip is requested, logical records are skipped until a logical end-of-file is encountered. This is repeated until the number of requested files has been skipped.

If a backward skip is requested, the code in the BCKREC portion of GINØ is used to position backwards one logical record. A test for load point is made. If yes, return is made. Otherwise, a logical end-of-file test is made. If no, the BCKREC is repeated. If yes, a test for number of requested files is made. If yes, return is given. Otherwise, the process is repeated.

8. EØF. If the file was not opened to write, an error message is generated. If CBP ≠ CLR, control words are placed to "close" the last logical record. A logical end-of-file is placed in the buffer. Return is made.

9. REWIND. The file is rewound, pointers are reset and return is given.

3.4.12.5 Design Requirements

1. GINØ is designed as an integral part of the GINØ collection of routines and is to be called only by XGINØ.

2. Since GINØ "remembers" the position of files when they are closed, any activity on a file outside of GINØ will likely be fatal.

3. Because of the packing used in the control word, the following maximums apply:

Max. number of blocks written on one file = $2^k - 1$ where k = number of bits in the computer word - 18.

Max. buffer size = 65535 words.

3.4.12.6 Diagnostic Messages

The following system fatal messages may be issued by GINØ:

3009

3029

3048

3049

3.4.12.7 Information Message

GINØ RECØVERY ATTEMPT ØN DATA BLØCK _____. EXPECTED BLØCK NØ. = _____. ACTUAL BLØCK NØ. = _____.

This message is issued prior to Message 3049. GINØ attempts to reposition the file. If the attempt fails, Message 3049 is issued, otherwise the problem proceeds.

3.4.13 ØPNCØR (Transmit Logical Records To/From Core Storage).

3.4.13.1 Entry Points: ØPNCØR, WRTCØR, RDCØR.

3.4.13.2 Purpose

To simulate the GINØ WRITE and READ calls providing the capability to write logical records of data in core storage and read logical records from core storage.

3.4.13.3 Calling Sequence

CALL ØPNCØR(BLØCK)

BLØCK - An array whose dimension is sufficient to hold a logical record to be written or read.

CALL WRTCØR(BLØCK,A,N,EØR)

BLØCK - The array where the logical record is to be written.

A - An array containing the data words to be written.

N - The number of data words to be written from A.

EØR - $\begin{cases} = 0, \text{ additional data will be written in the logical record via subsequent calls to WRTCØR.} \\ \neq 0, \text{ the current call is that last call for the current logical record.} \end{cases}$

CALL RDCØR($\$n_1$,$\$n_2$,BLØCK,A,N,EØR,M)

n_1 - FØRTRAN statement number defining the return taken in the event an end-of-file is encountered. This return is not possible from RDCØR but is provided in the calling sequence for compatibility with READ.

n_2 - FØRTRAN statement number defining the return taken in the event that the number of words requested to be read is not available in the record. In this case, M words are read and transmitted and the value M is returned to the user.

BLØCK - The array where the logical record is stored.

A - An array where the requested data words from the record will be stored.

N - The number of data words to be read.

EØR - { = 0, more data is to be read from the record via subsequent calls to RDCØR.
{ ≠ 0, the current call is the last call for the current logical record. Any remaining words in the logical record are to be skipped.

M - The number of words actually read in the event return to n_2 is taken.

The number of words available at BLØCK must be equal to (or greater than) the number of words in the logical record plus two.

ØPNCØR initializes a word pointer stored at BLØCK(1) to 1 and has no other function. The user may desire to perform this function himself with the statement BLØCK(1) = 1. This function must be accomplished prior to the first call to WRTCØR or RDCØR for each logical record.

3.4.13.4 Method

ØPNCØR. BLØCK(1) is set to one and return is made.

WRTCØR. The current pointer stored at BLØCK(1) is picked up. N words are transmitted from A to BLØCK beginning at the current pointer plus one. If EØR = 0, the new pointer is stored and return made. Otherwise, an end-of-record flag is stored in BLØCK following the last word written. The pointer (pointing to the flag) is stored and return made.

RDCØR. The current pointer is picked up from BLØCK(1). Words are transmitted from BLØCK beginning at the current pointer plus one to A until (1) the end-of-record flag is encountered in which case the actual number of words transmitted is stored in M and RETURN 2 is given, or (2) N words have been transmitted. If EØR = 0, the new pointer is stored and return is made. If EØR ≠ 0, BLØCK is searched until the flag is found, the pointer (pointing immediately prior to the flag) is saved and return is made.

3.4.13.5 Design Requirements

The flag value must be unique. Its value = (-16777215) must not be one of the data words written.

3.4.14 GØPEN (Short Form for Subroutine ØPEN With Header Record Processing).

3.4.14.1 Entry Point: GØPEN.

3.4.14.2 Purpose

To provide a short form (without the non-standard return of subroutine ØPEN) for opening a GINØ file, and to write a two-word header record if the data block is opened as output with rewind or to skip the header record if the data block is opened as input with rewind.

3.4.14.3 Calling Sequence

```
CALL GØPEN(FILE,BUFFER,ØPT)
```

where:

FILE = GINØ file name (see section 1.6.4.1).

BUFFER = GINØ buffer location.

ØPT = any of the open options permitted by subroutine ØPEN (see section 3.4.2).

3.4.14.4 Method

Open the file (subroutine ØPEN). If ØPT = input with rewind (0), skip the first record of the data block. If ØPT = output with rewind (1), write the two word BCD name of the data block as returned by subroutine FNAME.

3.4.14.5 Design Requirements

The data block must exist (must not be purged). If ØPT = input with rewind (0), the first record of the data block must be at least two words long. Subroutines used: ØPEN, READ, WRITE, FNAME, MESAGE.

3.4.14.6 Diagnostic Messages

If the data block is purged or if an end-of-file or end-of-record condition is encountered when reading the data block, subroutine MESAGE will be called with internal message numbers 1, 2, or 3, respectively (external message numbers are 3001, 3002 and 3003).

3.4.15 FREAD (Short Form for Subroutine READ).

3.4.15.1 Entry Point: FREAD.

3.4.15.2 Purpose

To provide a short form (without the non-standard returns of subroutine READ) of reading a GINØ file.

3.4.15.3 Calling Sequence

```
CALL FREAD(FILE,BLØCK,N,EØR)
```

where:

FILE = GINØ file name (see section 1.6.4.1).

BLØCK= array into which N items are to be read.

N = number of items to be read.

EØR = any end of record option permitted by subroutine READ (see section 3.4.5).

3.4.15.4 Method

Read the N items from FILE into BLØCK. If subroutine READ returns an end-of-file or end-of-record condition, subroutine MESAGE is called with a fatal error condition.

3.4.15.5 Design Requirements

In addition to those imposed by READ, there must be N items remaining in the record to be read. Subroutines used: READ, MESAGE.

3.4.15.6 Diagnostic Messages

Subroutine MESAGE may be called with internal message number 2 or 3 (external message numbers 3002,3003).

3.4.16 WRTTRL (Write Trailer).

3.4.16.1 Entry Points: WRTTRL, RDTRL.

3.4.16.2 Purpose

WRTTRL will pack six words of trailer information into three words and store them in the FIAT.

RDTRL will retrieve and unpack the trailer information.

3.4.16.3 Calling Sequence

CALL WRTTRL(FILBLK)

FILBLK(1) - GINØ file name (see section 1.6.4.1).

FILBLK(2-7) - Trailer information to be stored.

CALL RDTRL(FILBLK)

FILBLK(1) - GINØ file name.

FILBLK(2-7) - Storage space for trailer information.

3.4.16.4 Method

The index into the FIAT for the specified file is located in the FIST. The three packed words are stored in or retrieved from the FIAT. The information is also stored for all files equivalenced to the GINØ file name. For RDTRL, if the file is purged, FILBLK(1) is set negative.

3.4.16.5 Design Requirements

Each word of trailer information is assumed to be a positive integer less than $2^{16}-1$.

3.4.16.6 Diagnostic Messages

If the file did not exist in the FIST when WRTTRL was called, fatal error 3011 occurs.

3.4.17 FNAME (File Name).

3.4.17.1 Entry Point: FNAME.

3.4.17.2 Purpose

Given a GINØ file name, FNAME returns the two BCD words which describe the data block.

3.4.17.3 Calling Sequence

CALL FNAME(FILE,NAME)

FILE - GINØ file name (see section 1.6.4.1).

NAME(2) - Storage for the two BCD words.

3.4.17.4 Method

The GINØ file name is first located in the FIST. The index in the FIST is used to find the BCD descriptors in the FIAT. If the file does not exist in the FIST, "^(NØNE)^" is returned as the two words, ^ indicating a BCD blank.

3.4.18 CLSTAB (Close a GINØ File and Write a Non-zero Trailer).

3.4.18.1 Entry Point: CLSTAB.

3.4.18.2 Purpose

To close a GINØ file and generate a table trailer by calling WRTTRL.

3.4.18.3 Calling Sequence

CALL CLSTAB(FILE,ØPT)

where:

FILE = GINØ file number - integer - input.

ØPT = any close option permitted by subroutine CLØSE (see section 3.4.4) - integer - input.

3.4.18.4 Method

CALL CLØSE(FILE,ØPT)

Generate the table control block, ITABCB:

```
      ITABCB(1) = FILE
      ITABCB(7) = 1
      DØ 10 I   = 2,6
   10 ITABCB(I) = 0
```

CALL WRTTRL (ITABCB).

3.4.18.5 Design Requirements

Same as those for subroutines CLØSE and WRTTRL. Subroutines used: CLØSE, WRTTRL.

3.4.19 XRCARD (Executive Free-Field Card Data Conversion Routine)

3.4.19.1 Entry Point: XRCARD.

3.4.19.2 Purpose

To interpret NASTRAN free-field card input data as follows:

1. Identify BCD alpha and numeric data fields as they are converted and placed in the user's buffer;
2. Flag and output special data field delimiters;
3. Convert BCD numeric fields to binary integer or binary floating point;
4. Indicate when the data extends beyond one 72 column card.

3.4.19.3 Calling Sequence

```
CALL XRCARD(ØUTBUF,L,INBUF)
```

Where:

ØUTBUF	=	The buffer which is to contain the converted card image.
L	=	The length of ØUTBUF available to XRCARD.
INBUF	=	The buffer containing the card image to be converted.

3.4.19.4 Method

XRCARD's design is based on the necessity of having to function on a variety of computing machines having a variety of computer word structures, and a variety of differences in hollerith handling imposed by differing FØRTRAN compilers.

XRCARD analyzes the twenty hollerith words input through INBUF as follows:

Data Field Delimiters

Type A:

The following symbols signify the end of an alpha field or numeric field on the card. As these symbols are encountered, they will be flagged and placed in the output buffer to aid the user in identifying the data.

(LEFT PAREN
/ SLASH
= EQUAL

Type B:

The following symbol is identical to those listed above except that the symbol is not flagged or placed in the output buffer:

, COMMA

When successive type A or type B delimiters are encountered, a null field indication (two BCD blank words) is output. A null field is generated for each successive delimiter. A null field is also generated when a type A or type B delimiter is followed by a $ indicating the end of data condition.

Type C:

The following symbols are identical to the COMMA except that no null field indication is output when they are encountered in succession.

) RIGHT PAREN

^ BLANK

End of Data Indication

There are two means by which end-of-data may be specified on the card:

- The last data field ends in column 72, or is followed by blanks out through column 72; or a
- $ is encountered, after which comments may be included out to column 80.

Format of Output Data

A mode word, N, is placed in the output buffer to distinguish between BCD data and numeric data.

Numeric Mode Word: A new mode word is output each time a numeric field is converted and output. (All numeric mode words are negative).

N = -1 integer data (1 data word)
= -2 floating point single precision (1 data word)
= -4 floating point double precision (2 data words)

N indicates the type of numeric data and where to look for the next mode word.

Alpha Mode Word: When processing alpha data, only one mode word is output for successive alpha fields, i.e., an alpha mode word will never follow another alpha mode word.

N = The number of successive alpha fields encountered on the card. Each alpha field consists of two 4-character computer words (left adjusted). Thus N can be used to compute the location of the next mode word.

The type A delimiters are output as alpha data and are 'covered' by the alpha mode word. Since data output in the alpha mode must consists of two words a type A delimiter will appear as:

Word 1 = Delimiter flag, all bits of the word are on.

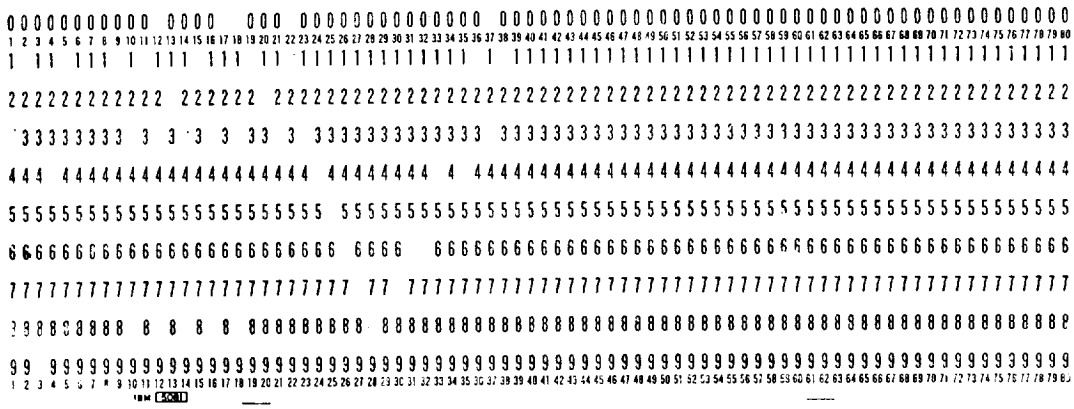
Word 2 = BCD delimiter, left adjusted, followed by BCD blanks.

End-of-Data: The end-of-data flag is placed last in the output buffer and appears in place of an expected mode word. There are two end-of-data flags:

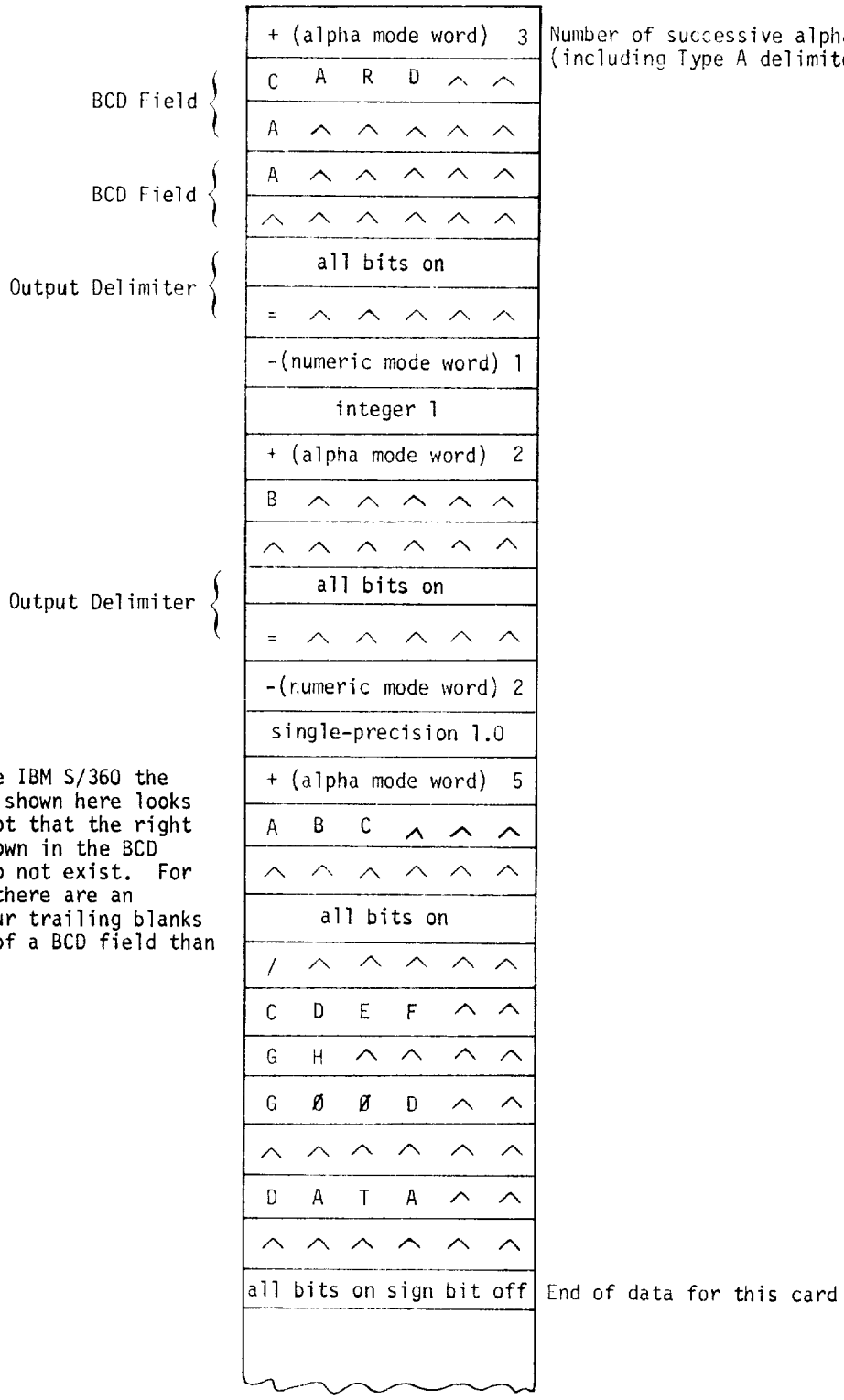
- A word with all bits off, indicating that more data is to follow on a continuation card.
- A word with all bits on except for the sign, indicating that no more data is to follow for this card type.

Sample Input Card

```
CARDA   A=1,B=1.0,ABC/CDEFGH GOOD DATA
```



Resulting Output Buffer for IBM 7094 or Univac 1108



NOTE: For the IBM S/360 the output buffer shown here looks the same except that the right two blanks shown in the BCD fields here do not exist. For the CDC 6600 there are an additional four trailing blanks in each word of a BCD field than shown here.

^ Indicates blank.

3.4.19.5 Design Requirements

An alpha field must be eight characters or less. Long alpha fields will be truncated to eight characters.

All data must be placed in card columns 1-72.

A data field may not be split between two cards.

The specification of all numeric data fields must conform to FØRTRAN IV standards.

If an error condition is encountered, e.g., data bad, XRCARD will write a message, turn on the NØGØ flag in /SYSTEM/, set the first word of ØUTBUF = 0, and make a normal return to the calling program.

3.4.20 RCARD (Fixed Field Card Data Conversion Routine).

3.4.20.1 Entry Point: RCARD.

3.4.20.2 Purpose

To interpret NASTRAN fixed-field (bulk data) card input as follows:

- Identify BCD alpha and numeric data fields as they are converted and placed in the users buffer; and

- Convert BCD numeric fields to binary integer or binary floating point.

3.4.20.3 Calling Sequence

CALL RCARD(ØUTBUF,FRMTBF,NFLAG,INBUF)

Where:

ØUTBUF = The buffer which is to contain the converted card image.

FRMTBF = A buffer which contains identification flags for the converted data in ØUTBUF.

NFLAG = Contains number of words returned in ØUTBUF.

INBUF = The buffer containing the card image to be converted.

Definition of Data Identification Flags Placed in FRMTBF

0 = output for a blank data field.

1 = output for an integer field.

2 = output for a floating point field.

3 = output for a BCD field.

4 = output for a double precision floating point field.

-1 = error.

3.4.20.4 Method

RCARD's design is based on the necessity of having to function on a variety of computing machines having a variety of computer word structures, and a variety of differences in Hollerith handling imposed by differing FØRTRAN compilers.

Twenty 4-Hollerith words are received by RCARD on any particular call to RCARD. RCARD first determines from field 1 (words 1 and 2) if the data card is a continuation card, and whether the

fields are single (2 words each) or double (4 words each) in length. Fields 2 through 9 (for single field cards) or 2 through 5 (for double field cards) are then considered one at a time. No consideration is made for the last field of any card (words 19 and 20).

3.4.20.5 Design Requirements

1. All BCD fields must begin with an alphabetic character.

2. All BCD fields are defined to be eight characters in length. Names with less than eight characters will be filled with BCD blanks.

3. When placed in the user output buffer, each BCD field will be divided into two four-character words (left adjusted) and stored in two successive locations of the output buffer. The remainder of the words is filled with BCD blanks.

4. Special characters are not to be used as part of a BCD field except for * and + in field 1 (column 1) which indicate a double field or single field (respectively) continuation card.

5. The data fields will be stored successively in the users output buffer as they are encountered in scanning the card image from left to right. The number of output core locations required per field type varies:

   a. Integer field = 1 core word (right adjusted).

   b. BCD field = 2 core words.

   c. Real single precision = 1 core word.

   d. Real double precision = 2 core words.

6. The card type field (field 1) of a continuation card will not be passed along to the user. Two zero words will replace the ID field in the output buffer. Thus the user can easily distinguish the difference between a continuation card and a new card type.

7. A check for bulk data card types SEQGP and SEQEP is made by RCARD. Fields 3, 5, 7, and 9 of these card types are processed by a special conversion.

The input within these special fields will be similar to the Dewey decimal notation and consists of a multiple digit integer and up to three single digit sub-integers; e.g., (354.1.2) and (267.5). The special fields will be converted to a single integer by dropping any decimal points and appending a number of zeros equal to three minus the number of decimal points in the original number; e.g., (354120) and (267500).

8. RCARD does not know the length of the users output buffer, therefore, no check is made for exceeding the length of the buffer. However, the number of data words placed in the output buffer will be specified in NFLAG.

9. Field 10 will not be passed along to the user.

3.4.20.6 Diagnostic Messages

Fields appearing to be incorrect to RCARD will cause a diagnostic to be written on the system output file followed by a card format heading, a card image echo, and an underlining of the field in question. Also, the /SYSTEM/ NØGØ flag is set .TRUE., a zero is placed in the output buffer for the field, and a -1 is placed in the format buffer for the field. RCARD will print diagnostics for all fields appearing incorrect and make a normal return.

3.4.21 TAPBIT (Tape Bit Test).

3.4.21.1 Entry Point: TAPBIT.

3.4.21.2 Purpose

To examine the tape bit for a permanent GINØ file to determine the existence of a physical tape for that file.

3.4.21.3 Calling Sequence

```
IF (TAPBIT(FILE)) GØ TØ ...
```

FILE is the GINØ file name (one of 'PØØL', 'ØPTP, 'NPTP', 'UMF ', 'NUMF', 'PLT1', 'PLT2', 'INPT').

3.4.21.4 Method

The permanent FIST is searched and the tape bit in the corresponding FIAT entry is examined. If the bit is on (indicating the presence of a physical tape), TAPBIT will be set .TRUE.. Otherwise it will be set .FALSE..

3.4.21.5 Design Requirements

The type of TAPBIT must be declared LØGICAL.

3.4.21.6 Diagnostic Messages

A fatal call to MESAGE occurs if a GINØ file name other than those listed is used.

3.4.22 PEXIT (Problem Exit).

3.4.22.1 Entry Point: PEXIT.

3.4.22.2 Purpose

To terminate the program.

3.4.22.3 Calling Sequence

CALL PEXIT.

3.4.22.4 Method

The diagnostic message queue is checked and if not empty the message writer MSGWRT is called. If the checkpoint flag is set a card is punched indicating the end of the restart checkpoint dictionary. The system output buffers are flushed and then the job is terminated.

3.4.22.5 Design Requirements

PEXIT must have access to the FØRTRAN I/Ø routines.

3.4.23 TMTØGØ (Time-To-Go).

3.4.23.1 Entry Point: TMTØGØ.

3.4.23.2 Purpose

Computes the running time remaining for this NASTRAN problem.

3.4.23.3 Calling Sequence

CALL TMTØGØ (TIME)

TIME = Remaining time in integer seconds.

3.4.23.4 Method

During NASTRAN problem initialization, one system cell is set to the problem starting time (PSTART) while another is set to the maximum running time (MXTIME) contained on the Executive Control Deck TIME card. TIME-TØ-GØ is then found by reading the clock (NØW) and solving the following:

TIME-TØ-GØ = MXTIME - (NØW - PSTART).

The CPU clock is utilized on all machines except the IBM 7094 where none is available.

3.4.24 PAGE (Page Heading).

3.4.24.1 Entry Points: PAGE, PAGE1, PAGE2

3.4.24.2 Purpose

To provide a standard page heading for NASTRAN output.

3.4.24.3 Calling Sequence

```
CALL PAGE
CALL PAGE1
CALL PAGE2(N)

CØMMØN/SYSTEM/XXX,ØTPE,SPACE(6),IPAGE,LINE,ITLINE,MAXLIN,DATE(3)
```

ØTPE - System output unit - integer.

IPAGE - Current page number - increased by 1 on each call to PAGE.

LINE - Number of data lines on previous page - LINE is set to zero by PAGE.

ITLINE - Total number of lines of printout in run - ITLINE = ITLINE + LINE.

MAXLIN - Maximum number of data lines allowed - if ITLINE > MAXLIN, PEXIT will be called.

DATE(3)- Today's date: month, day, year - integer.

N - Number of lines to be written - integer - input.

```
CØMMØN/ØUTPUT/TITLE(32),SUBTIT(32),LABEL(32),HEAD1(32),HEAD2(32),HEAD3(32)
```

3.4.24.4 Method

PAGE writes a standard 6 line heading from TITLE, SUBTIT, LABEL, HEAD1, HEAD2, HEAD3.

PAGE1 writes only the first 3 lines of a standard header.

PAGE2 restores the page if N lines will not fit on the current page.

3.4.24.5 Design Requirements

ITLINE must be less than MAXLIN. PAGE must have access to the FØRTRAN I/Ø routines.

3.4.25 MESAGE (Message).

3.4.25.1 Entry Point: MESAGE.

3.4.25.2 Purpose

To queue nonfatal messages during the execution of a module; and for fatal messages give a core dump (CALL PDUMP), print the message queue (CALL MSGWRT), and call PEXIT.

3.4.25.3 Calling Sequence

CALL MESAGE(NØ,PARM,NAME)

Where

NØ = Internal message number. NØ positive defines the message as nonfatal; NØ negative defines the message as fatal.

PARM = Parameter used in the printed message (usually the GINØ file number)

NAME(2) = Two words used in the printed message (usually two BCD words containing the name of the subroutine calling MESAGE).

3.4.25.4 Method

Non-fatal messages are queued in common block /MSGX/ until the maximum number is reached. All non-fatal messages after this are lost. When a fatal message is encountered, it is queued and appropriate action taken to terminate the run.

3.4.25.5 Design Requirements

The size of common block /MSGX/ limits the number of messages stored.

3.4.26 MSGWRT (Message Writer).

3.4.26.1 Entry Point: MSGWRT.

3.4.26.2 Purpose

To print NASTRAN error messages on the system output file.

3.4.26.3 Calling Sequence

```
CALL MSGWRT

CØMMØN/MSGX/N,M,MSG(4,10)
```

where:

N - is the total number of messages to be printed.

M - maximum number of messages that can be queued by subroutine MESAGE in the array MSG.

MSG - array where message parameters are queued.

MSG(1,I) - the internal message number of the I^{th} message.

MSG(2,I) - if |MSG(1,I)| ≠ 30, MSG(2,I) is a GINØ file number.
If |MSG(1,I)| = 30, then MSG(2,I) is an internal message number and USRMSG is called.

MSG(3,I), MSG(4,I) = parameters for the I^{th} message.

3.4.26.4 Method

The internal message number, M(1,I), if not equal to 30 in absolute value, is used by MSGWRT to print out the error message along with external message number, which is 3000 plus the internal message number. If the internal message number, M(1,I), is 30, subroutine USRMSG is called.

3.4.26.5 Design Requirements

External message numbers output by MSGWRT at present are 3001 through 3057.

MSGWRT is called only by MESAGE.

3.4.27 USRMSG (User Message Writer).

3.4.27.1 Entry Point: USRMSG.

3.4.27.2 Purpose

To print most NASTRAN user error messsges on the system output file.

3.4.27.3 Calling Sequence

```
CALL USRMSG(I)

CØMMØN/MSGX/N,M,MSG(4,10)
```

where:

I - Pointer into the MSG array.

N - Not used in USRMSG.

M - Not used in USRMSG.

MSG(1,I) - If |MSG(1,I)| = 30, MSGWRT will call USRMSG.

MSG(2,I) - Used by USRMSG as the internal message number.

MSG(3,I), MSG(4,I) - Parameters for the I^{th} message.

3.4.27.4 Method

USRMSG will print appropriate error message along with external message number, which is 2000 plus internal message number.

3.4.27.5 Design Requirements

External message numbers output by USRMSG at present are: 2001--2137.

USRMSG is called only by MSGWRT.

3.4.28 MATDUM (Matrix Dump (Print) Routine).

3.4.28.1 Entry Point: MATDUM.

3.4.28.2 Purpose

To print a general NASTRAN matrix.

3.4.28.3 Calling Sequence

CALL MATDUM(FILEA)

FILEA - Seven-word array (matrix control block) - integer

Word	
1	GINØ name
2	Number of columns
3	Number of rows
4	Form of matrix
5	Type of matrix
6	Maximum number of non-zero terms in any column
7	Undefined

3.4.28.4 Method

The non-zero terms of each column are unpacked and printed.

If the matrix control block does not contain legal values the table printer (see section 3.4.29) is called.

3.4.28.5 Design Requirements

Open core at /TABPRX/.

MATDUM must hold the non-zero band of the matrix in this area.

Subroutine TABPRT and the FØRTRAN I/Ø routines must be available to MATDUM.

3.4.29 TABPRT (Table Printer).

3.4.29.1 Entry Point: TABPRT.

3.4.29.2 Purpose

To print any NASTRAN Data Block (especially tables).

3.4.29.3 Calling Sequence

CALL TABPRT(FILEN)

FILEN - GINØ name of data block - integer - input.

3.4.29.4 Method

Each word is read, identified as to type -- integer, BCD, or real number and printed 10 characters per word, 10 numbers per line. Note that the identification method varies from machine to machine and is not 100% certain, i.e., certain words may be misidentified.

3.4.29.5 Design Requirements

Open core at /TABPRX/.

Double precision numbers will not be correctly interpreted on the Univac 1108.

3.4.30 PRELØC (Position Data Block to Requested Record).

3.4.30.1 Entry Points: PRELØC, LØCATE.

3.4.30.2 Purpose

To provide a convenient means of locating data records in data blocks output by the Input File Processor (IFP).

3.4.30.3 Calling Sequence

CALL PRELØC($n,BUFF,NAME)

n - FØRTRAN statement number defining return taken in the event NAME is not in the FIST (i.e., data block is purged).

BUFF - An array whose dimension equals the contents of the first word of /SYSTEM/ plus one. Used as a GINØ buffer by PRELØC and LØCATE.

NAME - GINØ file name of data block to be read (integer).

CALL LØCATE($n,BUFF,ID,IDX)

n - FØRTRAN statement number defining return taken in the event that the requested record (defined by ID) is not present in the data block.

BUFF - The same BUFF assigned when PRELØC was called.

ID - The address of a two-word array. The first word is the integer record identification and the second word is the bit position in the trailer for the data block where the presence or absence of the record is defined.

IDX - The contents of the third word of the record found will be stored in IDX (internal card number generated by IFP).

Notes:

1. If the data block is not purged, PRELØC will open the file and skip the header record.

2. If the requested record is not present (as determined by the appropriate trailer bit), no I/Ø activity will occur. Otherwise, LØCATE will position the file to read the first data entry of the requested record (i.e., after the 3-word header for the record). See 2.3.2 for format of records and trailer.

3. If the user does not read all data in a record and he wishes to use LØCATE to find another record, he should use FWDREC to skip the remainder of the current record prior to calling LØCATE.

4. For optimum efficiency in processing a data block, the user should call LØCATE in the order in which the records appear on the data block, i.e. NASTRAN collating order.

3.4.30.4 Method

PRELØC stores NAME in BUFF(1) and then calls ØPEN using BUFF(2) as the buffer address. If the data block is purged, the non-standard return is given to the user. Otherwise, FWDREC is called to skip the header record and return is made to the user. LØCATE calls RDTRL to read the data block trailer. The bit position identified by ID(2) is tested using ANDF. If zero, the non-standard return is given. Otherwise, three words from the file are read. If the first word equals ID(1), IDX is set to the third word and return is made. Otherwise, the first word is saved and the remainder of the record is skipped. The first three words of each successive record are read and the test for match on first word is made until (1) an end-of-file occurs in which case the file is rewound, the header record skipped and the process is continued, (2) a match is found in which case IDX is set and return is given or (3) a match with the first record read is found in which case the record is skipped, a warning message is queued and the non-standard return is given.

3.4.30.5 Diagnostic Messages

The following messages may be issued by PRELØC:

2072

3002

3003

3.4.31 SØRT (Sort a Table).

3.4.31.1 Entry Point: SØRT.

3.4.31.2 Purpose

To sort a core contained table, or to sort a logical record from a specified input file, on a specified keyword in each entry.

3.4.31.3 Calling Sequences

To sort a core contained table:

CALL SØRT(0,0,NWDS,KEYWD,TABLE,NTABLE)

NWDS - The number of words in each entry of the table. Restriction: NWDS ≤ 20.

KEYWD - The word position within each entry on which the sort is to take place.

TABLE - Address where the table is stored.

NTABLE - Total number of words in the table (NTABLE must be an integral multiple of NWDS).

To sort a logical record:

CØMMØN/SETUP/NFILE(6),BUF

CALL SØRT(INPFL,ØUTFL,NWDS,KEYWD,BLØCK,NBLØCK)

NFILE - The first three words must be set by the user prior to CALL SØRT with the GINØ file names of three scratch files for use by SØRT. Upon return to the user, NFILE(6) will contain the GINØ file name of the file containing the sorted record.

BUF - If INPFL = ØUTFL, then BUF points to an area in BLØCK where a GINØ buffer is available for SØRT, i.e., BLØCK(BUF) is the buffer address. Restriction: BUF > NBLØCK.

INPFL - GINØ file name of data block containing the logical record to be sorted.

ØUTFL - GINØ file name of data block where the sorted record is to be written. If ØUTFL = 0, the sorted record will remain on NFILE(6).

NWDS - The number of words in each entry of the record. Restriction: NWDS ≤ 20.

KEYWD - Defined as above.

BLØCK - An area in core to be used by SØRT to perform the sort phase.

NBLØCK - The number of computer words available at BLØCK.

Notes:

1. INPFL must be opened and positioned to the logical record by the user prior to entry to SØRT. The file is not closed by SØRT.

2. If ØUTFL ≠ 0, this file must be opened and positioned by the user prior to entry to SØRT. The file is not closed by SØRT.

3. If INPFL = ØUTFL, the file is closed by SØRT, opened to write with rewind, and the sorted logical record is written as the first logical record on the file. The file is not closed by SØRT.

4. NFILE(6) is always closed with rewind.

3.4.31.4 Method

1. CØRE SØRT. The method used is a shuttle exchange or bubble sort which is optimum for data which is nearly in sort. The method is as follows:

a. The key words of two successive entries are compared. If currently in sort, the process is repeated. If not,

b. A search toward the beginning of the table is made to determine the position of the out-of-sort entry.

c. From this position, the table is shifted one entry and the out-of-sort entry is inserted at its proper position.

d. If the last pair of entries have not been analyzed, the process returns to step (a). Otherwise the sort is complete.

2. FILE SØRT. One GINØ buffer is allocated at the end of BLØCK and a scratch file is opened to write. As many entries as can be held in the remaining core in BLØCK are read and sorted using the algorithm above. The sorted data is written as a logical record on the scratch file. This process is repeated until all data in the input record has been read and the sorted

strings written on the scratch file. If only one such sort was required, the sort is complete except for copying onto ØUTFL if requested. Otherwise, an optimum distribution of sorted records on two scratch files is computed using a Fibonacci sequence. The sorted strings are redistributed between two scratch files and the merge phase is entered. The two scratch files are read one entry at a time, merged, and new sorted entries written on a third scratch file. Note that, using the Fibonacci sequence, one of the files containing sorted strings will have a greater number of strings (records) than the other. On each pass in the merge phase, the merge occurs until the file with fewer strings is exhausted. At this point, the merged file becomes the file with the larger number of sorted strings, the previous larger file becomes the file with the fewer strings, and the previous file with fewer strings (which was exhausted) becomes the file onto which the merged strings are written. The process continues until the sort is complete. The resulting sorted record is copied onto ØUTFL if requested.

3.4.31.5 Design Requirements

The number of words per entry may not exceed 20. (A change in the dimension of the local variable TEMP may be made to relax this restriction.)

The amount of core available at BLØCK must be at least one GINØ buffer plus 2*NWDS during the core sort phase and three GINØ buffers plus 2*NWDS during the merge phase.

The core table or logical record to be sorted must contain an integral number of entries.

3.4.31.6 Diagnostic Messages

The following messages may be issued by SØRT:

3001

3002

3008

3.4.32 GMMATD (General Matrix Multiply and Transpose - Double Precision).

3.4.32.1 Entry Point: GMMATD.

3.4.32.2 Purpose

To perform any one of the following matrix operations:

$$[A]\,[B] = [C] \tag{1}$$

$$[A]^T\,[B] = [C] \tag{2}$$

$$[A]\,[B]^T = [C] \tag{3}$$

$$[A]^T\,[B]^T = [C] \tag{4}$$

$$[A]\,[B] + [D] = [C] \tag{5}$$

$$[A]^T\,[B] + [D] = [C] \tag{6}$$

$$[A]\,[B]^T + [D] = [C] \tag{7}$$

$$[A]^T\,[B]^T + [D] = [C] \tag{8}$$

where [A], [B], [C], and [D] are real double precision matrices. This routine is used for small in-core matrices, in non-NASTRAN packed format, in such modules as SMA1, SMA2, SMA3 and DSMG1.

3.4.32.3 Calling Sequence

CALL GMMATD(A,IRØWA,ICØLA,MTA,B,IRØWB,ICØLB,MTB,C)

A - A real double precision matrix of IRØWA rows and ICØLA columns stored in the singly dimensioned double precision variable A.

N.B. A must be stored by rows. For example, if

$$[A] = \begin{bmatrix} 1.0 & 4.0 \\ 2.0 & 5.0 \\ 3.0 & 6.0 \end{bmatrix},$$

then the matrix must be stored in the FØRTRAN double precision array A as follows:

A(1) = 1.0

A(2) = 4.0

A(3) = 2.0

A(4) = 5.0

A(5) = 3.0

A(6) = 6.0

(A is input only).

IRØWA - number or rows of [A] - input.

ICØLA - number of columns of [A] - input.

MTA - Flag used to determine if [A] is to be transposed and to determine if the output matrix, [C], is to be zeroed out; that is, to determine if a matrix product only, of the form [A] [B] = [C], will be performed or if a product and (in effect) a sum, of the form [A] [B] + [D] = [C], will be performed.

1. If MTA = 0, then [A] is not transposed and hence either Equation (1) or (3) will be performed, depending upon MTB.

   If MTA = +1 then [A] is transposed and hence either Equation (2) or (4) will be performed, depending upon MTB.

   MTA is input only.

2. If MTA is less than zero, [C] is not zeroed out. Hence the routine, in this case, computes

   $[A][B] + [D] = [C]$ if MTA = -2 and MTB = 0.

   $[A][B]^T + [D] = [C]$ if MTA = -2 and MTB = 1.

   $[A]^T[B] + [D] = [C]$ if MTA = -1 and MTB = 0.

   $[A]^T[B]^T + [D] = [C]$ if MTA = -1 and MTB = 1.

   (see MTB definition below)

   where D is a real double precision matrix of IRØWA rows and ICØLB columns if MTA = -2 and D is ICØLA x ICØLB if MTA = -1. D must be stored row-wise at the location of C by the calling program.

B - real double precision matrix, stored row-wise. See comments for A above - input.

IRØWB - the number or rows of [B] - input.

ICØLB - the number of columns of [B] - input.

MTB - Transpose flag for [B]. If MTB = 0, [B] is not transposed. If MTB = 1, [B] is transposed. Note that MTA and MTB are independent and that only MTA controls whether or not

[C] will be zeroed out. MTB is input only.

C - real double precision matrix. Input (if MTA < 0) and output.

Examples on the use of the routine:

1. If [A] is 3x3 and [B] is 3x1 and [C] = [A] [B] is desired then:

   CALL GMMATD(A,3,3,0,B,3,1,0,C). [C] is 3x1.

2. If [A] is nx1 and [B] is nx1 and the dot product is desired ($[A]^T$ [B]) then:

   CALL GMMATD(A,N,1,1,B,N,1,0,C). [C] is 1x1, a scalar.

3. Compute [C] = $([X] [Y])^T$ where [X] is 5x4 and [Y] is 4x7:

   CALL GMMATD(Y,4,7,1,X,5,4,1,C). C is 7x5.

4. Compute D = [A] $[B]^T$ + [C] where [A], [B] and [C] are 3x3:

```
      DO 10 I = 1, 9
   10 D(I) = C(I)
      CALL GMMATD(A,3,3,-2,B,3,3,1,D).
```

3.4.32.4 Method

The first phase of the subroutine sets up integer loop limits which are functions of the two transpose flags. If MTA is not less than zero, the C array is zeroed out. Then the classical mathematical definitions of the above matrix products are carried out.

3.4.32.5 Design Requirements

The orders of the [A] and the [B] matrices in combination with the transpose flags must define a conformable matrix product.

3.4.32.6 Diagnostic Messages

The subroutine examines the tranpose flags in combination with the orders of the matrices to make sure that a conformable matrix product is defined by this input data. This test clearly is made for purposes of calling routine checkout only. No tests are made, nor can they be made, to insure that the calling routine has provided sufficient storage for arrays. If a conformable matrix product is not defined by the input arguments, fatal error message 2021 is printed.

3.4.33 GMMATS (General Matrix Multiply and Transpose - Single Precision).

3.4.33.1 Entry Point: GMMATS.

3.4.33.2 Purpose

To perform any one of the following matrix operations:

$$[A]\,[B] = [C] \tag{1}$$

$$[A]^T\,[B] = [C] \tag{2}$$

$$[A]\,[B]^T = [C] \tag{3}$$

$$[A]^T\,[B]^T = [C] \tag{4}$$

$$[A]\,[B] + [D] = [C] \tag{5}$$

$$[A]^T\,[B] + [D] = [C] \tag{6}$$

$$[A]\,[B]^T + [D] = [C] \tag{7}$$

$$[A]^T\,[B]^T + [D] = [C] \tag{8}$$

where [A], [B], [D] and [C] are real single precision matrices. This routine is used for small in-core matrices in non-NASTRAN packed format in such modules as SDR2 and PLA3 and in the utility routine PREMAT.

3.4.33.3 Calling Sequence

```
CALL GMMATS(A,IRØWA,ICØLA,MTA,B,IRØWB,ICØLB,MTB,C)
```

This routine is exactly the same as subroutine GMMATD except that GMMATD operates on real double precision matrices, while GMMATS operates on real single precision matrices. See subroutine description for GMMATD (see section 3.4.32) for details on subroutine arguments, method, design requirements and diagnostic messages.

3.4.34 INVERD (Double Precision In Core Inverse Routine).

3.4.34.1 Entry Point: INVERD.

3.4.34.2 Purpose

To compute the inverse of a real double precision matrix [A] and on option to solve the matrix equation [A] [X] = [B]. This routine is used to invert small in-core double precision matrices in non-NASTRAN packed format and is used as a utility routine in such modules as SMA1, SMA3 and DSMG1.

3.4.34.3 Calling Sequence

CALL INVERD (NDIM,A,N,B,M,DETERM,ISING,INDEX)

NDIM - The actual row dimension of the doubly subscripted arrays A and B in the calling program - integer - input.

A - The square matrix to be inverted. $[A]^{-1}$ upon return from INVERD is stored at A. Double precision - input and output

N - The order of the matrix being inverted (the size of the upper left hand corner actually being inverted). $N \leq NDIM$ - integer - input.

B - The column(s) of constants in the above equation. If [A] is to be inverted, then B is a dummy argument. The solution matrix [X] is returned at B. Double precision - input and output.

M - The number of columns of constants. If $M \leq 0$, $[A]^{-1}$ is computed - integer - input.

DETERM - The determinant of [A]. Double precision - output.

ISING - Singularity indicator. If [A] is non-singular, ISING is set to 1; if [A] is singular, ISING is set to 2 - integer - output.

INDEX - Doubly subscripted array of row dimension N and column dimension 3 used for the row and column interchanges - integer - internal working storage.

3.4.34.4 Method

The classical Gauss-Jordan method with full row and column interchanges is used. All arithmetic operations are double precision.

3.4.35 INVERS (Single Precision In Core Inverse Routine).

3.4.35.1 Entry Point: INVERS.

3.4.35.2 Purpose

To compute the inverse of a real single precision matrix [A] and on option to solve the matrix equation [A] [X] = [B]. This routine is used to invert small in-core single precision matrices in non-NASTRAN packed format and is used as a utility routine in such modules as SDR2.

3.4.35.3 Calling Sequence

CALL INVERS (NDIM,A,N,B,M,DETERM,ISING,INDEX)

This routine is exactly the same as subroutine INVERD except that INVERD operates on real double precision matrices, while INVERS operates on real single precision matrices. All arithmetic operations are single precision. DETERM is real single precision. See subroutine description for INVERD (see section 3.4.34) for details on subroutine arguments and method.

3.4.36 PREMAT (Material Property Utility).

3.4.36.1 Entry Points: PREMAT, MAT.

3.4.36.2 Purpose

To provide a utility routine for obtaining material properties used by structural element subroutines. The first entry point, PREMAT, is called once by a module for initialization purposes, and then MAT can be called by the module's element subroutines repeatedly to fetch required material properties.

3.4.36.3 Calling Sequence

CALL PREMAT (Z,ZZ,BFR,N1MAT,N2MAT,MPTF,DITF)

Z - Integer array of open core given to the subroutine to store the material properties and the direct input tables - input and output.

ZZ - Same address as Z. Used as real in this routine - input and output.

BFR - A GINØ buffer (plus one cell) used by subroutine PRELØC as a buffer - input only.

N1MAT - The length of open core, the Z array, given to PREMAT and MAT - integer - input only.

N2MAT - The length of open core used by PREMAT and MAT - integer - output only.

MPTF - GINØ file number of the Material Properties Table (MPT) data block - input only.

DITF - GINØ file number of the Direct Input Tables data block. If DITF is negative, the routine assumes that the calling module is a Piecewise Linear Analysis module which implies material properties cannot be temperature dependent and that MATS1 cards are to be read.

PREMAT uses the 10^{th} word of /SYSTEM/ which is the temperature set identification number for material properties chosen by the user in his Case Control Deck. PREMAT also uses /NAMES/ for various GINØ options.

CALL MAT (ELEMID)

ELEMID - Integer element identification number; used only for diagnostic messages (see below) - input and output.

CØMMØN/MATIN/MATID,INFLAG,TEMP,PLAARG,SINTH,CØSTH

MATID - Material property identification number - integer - input.

INFLAG - Integer input flag which determines which sets of input data cards, MAT1, MAT2, or MAT3, the routine will search in order to find MATID. Also INFLAG determines in what format the output will be placed in the MATØUT common block. Currently INFLAG may assume the values 1 through 7 defined as follows:

INFLAG = 1 -- The material properties corresponding to the MATID are output in "MAT1" or isotropic material format (see /MATØUT/ below). One dimensional elements such as RØD, BAR, SHEAR etc. require isotropic materials. If the MATID is not found among all the MAT1 material cards read by PREMAT, a fatal error occurs.

INFLAG = 2 -- If INFLAG = 2, the material properties corresponding to the MATID are output in "MAT2" or anisotropic material format. Two-dimensional elements such as TRMEM, TRIA1, QDPLT, QUAD1 etc. may use isotropic or anisotropic materials. First, the routine will try to find the MATID among the MAT1 cards. If it is found among the MAT1 cards, the variables E (modulus of elasticity), ν (Poisson's ratio) and G (shear modulus) are used to construct the 3x3 symmetric matrix $[G_e]$ needed by two-dimensional elements, and the matrix is stored in /MATØUT/:

$$[G_e] = \begin{bmatrix} \frac{E}{1-\nu^2} & \frac{\nu E}{1-\nu^2} & 0 \\ \frac{\nu E}{1-\nu^2} & \frac{E}{1-\nu^2} & 0 \\ 0 & 0 & G \end{bmatrix}$$

If the MATID is not found among the MAT1 cards, the MAT2 cards are searched. If the MATID is not found among the MAT2 cards a fatal error occurs. If it is found, $[G_m]$, the 3x3 symmetric matrix input on the MAT2 card, is transformed by the matrix equation $[G_e] = [U]^T[G_m][U]$ and $\{\alpha\} = [U]\{\alpha_m\}$, where $\{\alpha_m\}$ is the temperature expansion coefficient vector input on a MAT2 card. [U] is a function of sin θ and cos θ (see SINTH and CØSTH below).

$$[U] = \begin{bmatrix} \cos^2\theta & \sin^2\theta & \cos\theta\sin\theta \\ \sin^2\theta & \cos^2\theta & -\cos\theta\sin\theta \\ -2\cos\theta\sin\theta & 2\cos\theta\sin\theta & (\cos^2\theta-\sin^2\theta) \end{bmatrix}$$

INFLAG = 3 -- If INFLAG = 3, it implies the inverse of the symmetric 2x2 transverse shear matrix J will be stored in locations 16, 17 and 18 of /MATØUT/. There are two cases: (1) the current MATID is not equal to the most recent MATID, MATIDØ, and (2) the current MATID is equal to the most recent MATID.

1. If the current MATID is not equal to the most recent material identification number (MATIDØ), the MAT1 cards are searched. If the MATID is found among the MAT1 cards, then locations 16,17 and 18 of /MATØUT/ are set to G, 0.0 and G respectively, where G is the shear modulus. If the MATID is not found among the MAT1 cards, the MAT2 cards are searched. If the MATID is not found among the MAT2 cards, a fatal error occurs. If it is found among the MAT2 cards, locations 16, 17 and 18 are set to zero.

2. The current MATID is equal to the most recent MATID. If INFLGØ, the most recent INFLAG is not 2, this is the same as case (1). If it is 2, then (a) if the MATID was found on a MAT1 card, locations 16, 17 and 18 are set to G, 0.0 and G respectively; or (b) if the MATID was found on a MAT2 card, locations 16, 17 and 18 are set to 0.0.

INFLAG = 4 -- If INFLAG is 4, this implies that only the density of the material, RHØ, will be returned in /MATØUT/ and this in the first location. The MATID can be either on a MAT1 or MAT2 card. If the MATID cannot be found among all MAT1 and MAT2 cards, a fatal error occurs.

INFLAG = 5 -- INFLAG = 5 is reserved for use only by module PLA1. This option determines if the MATID is such that E, the modulus of elasticity, is defined as stress dependent by MATS1 and TABLES1 cards. If it is stress dependent, INDSTR, equivalenced to the first word of /MATØUT/, is set to +1. If not stress dependent, INDSTR is set to 0. Only MAT1 cards are admissible for INFLAG = 5.

INFLAG = 6 -- INFLAG = 6 is reserved for use by modules PLA3 and PLA4. The fourth word of /MATIN/, PLAARG (see below), is strain and is used as the independent variable in a table look-up for stress, which is stored in the first word of /MATØUT/. Only MAT1 cards are searched to match the input MATID.

INFLAG = 7 -- INFLAG 7 implies that the material properties corresponding to the MATID will be output in MAT3 or orthotropic material format. Currently only the axisymmetric elements TRIARG, TRAPRG and TØRDRG use this option. If the MATID is found in the MAT1 set, the data are stored in MAT3 format. If not found in the MAT1 set, the MAT3 set is searched. If not found here, a fatal error exists.

INFLAG = 8 -- INFLAG = 8 is used only by two-dimensional element subroutines in modules PLA3 and PLA4. The fourth word of /MATIN/, PLAARG (see below), is stress (σ) and is used as the ordinate in an inverse interpolation table look-up to obtain the abscissa which is strain (ε).

If either: a) the ordinate is in the range of the piecewise linear function defined by the table on a TABLES1 bulk data card, or b) the ordinate is greater than the maximum (which is also the last) ordinate in the table but the slope of the line segment joining the last two points of the table is nonzero, then the second word of /MATØUT/ is set to zero and the abscissa, obtained by inverse linear interpolation or extrapolation, is stored in the first word of /MATØUT/. If either: a) the ordinate is less than the minimum (which is also the first) ordinate in the table, or b) the ordinate is greater than the maximum ordinate in the table and the slope of the line segment joining the last two points of the table is zero, then the integer "1" is stored in the second word of /MATØUT/ (and the first word of /MATØUT/ is set to zero). Only MAT1 cards are searched to match the input MATID.

TEMP - Average element temperature. Used as the independent variable in a table look-up when it is determined that a material property is temperature dependent. Not used when INFLAG = 5 or 6.

PLAARG - Element strain. Used as the independent variable in a table look-up when E, the modulus of elasticity, is defined as the first derivative of a strain-stress curve. Used only in the Piecewise Linear Analysis Rigid Format and only by modules PLA3 and PLA4.

SINTH - Sine of the material property orientation angle. Used only when INFLAG = 2 and the MATID is found among the MAT2 cards. Used to construct the [U] matrix referenced above.

CØSTH - Cosine of the material property orientation angle. The comments on SINTH, above, also apply here.

CØMMØN/MATØUT/ - (Output Common Block). Length 20 words. Depending upon the values of INFLAG, the output common block is defined variously as follows:

1. MAT1 Format (INFLAG = 1)

Word	Symbol	Definition
1	E	Young's modulus (modulus of elasticity)
2	G	Shear Modulus
3	ν	Poisson's ratio
4	ρ	Density
5	α	Thermal expansion coefficient
6	T_o	Thermal expansion reference temperature
7	g_e	Structural element damping coefficient
8	σ_t	Stress limit for tension
9	σ_c	Stress limit for compression
10	σ_s	Stress limit for shear
11-20	-	Undefined

2. MAT2 Format (INFLAG = 2)

Word	Symbol	Definition
1	G11	The 3x3 symmetric material property matrix
2	G12	
3	G13	
4	G22	
5	G23	
6	G33	
7	RHØY	Density
8	ALPH1	Thermal expansion coefficient vector
9	ALPH2	
10	ALPH12	
11	TØY	Thermal expansion reference temperature
12	GEY	Structural element damping coefficient
13	SIGTY	Stress limit for tension
14	SIGCY	Stress limit for compression
15	SIGSY	Stress limit for shear
16-20	-	Undefined

3. Transverse Shear Inverse Matrix (INFLAG = 3)

Word	Symbol	Definition
1-15	-	Unchanged
16	J11	The 2x2 symmetric inverse of the transverse shear matrix
17	J12	
18	J22	
19-20		Undefined

4. RHØ Only Format (INFLAG = 4)

Word	Symbol	Definition
1	RHØ	Density
2-20	-	Undefined

5. PLA1 Use Only (INFLAG = 5)

Word	Symbol	Definition
1	INDSTR	Stress dependent flag
2-20	-	Undefined

6. Stress Functional Value (INFLAG = 6)

Word	Symbol	Definition
1	PLAANS	Value of stress (σ) as a function of ε (strain)
2-20	-	Undefined

7. MAT3 Format (INFLAG = 7)

Word	Symbol	Definition
1	EX3	Young's Moduli x, y and z directions
2	EY3	
3	EZ3	
4	NUXY3	Poisson's ratios. Coupled strain ratios in the xy, yz, and zx directions
5	NUYZ3	
6	NUZX3	
7	RHØ3	Density
8	GXY3	Shear moduli
9	GYZ3	
10	GZX3	
11	AX3	Thermal expansion coefficients
12	AY3	
13	AZ3	
14	TREF3	Thermal expansion reference temperature
15	GE3	Structural element damping coefficient
16-20	-	Undefined

8. Strain Functional Value (INFLAG = 8)

Word	Symbol	Definition
1	PLAANS	Value of strain (ε) as an inverse function of stress (σ)
2	ICELL2	= 0 if the input stress is in the range of the function = 1 if the input stress is outside the range of the function
3-20		Undefined

3.4.36.4 Method

1. PREMAT: All the MAT1, MAT2 and MAT3 cards are read from the MPT data block into open core so that each card is assigned 1 + 3*N words of core where N, a function of the card type, is the number of material property data items on that card type. The first word is the material identification number and each material property is allocated 3 words: the first the input material property; the second a table (function) number which gives this material property as a function of temperature; the third a table number which gives this material property as a function of stress. Initially words 2 and 3 are set to zero. Although the third word is currently used only for MAT1 cards and for E, the modulus of elasticity, on that card, future development may make use of a more general application of stress dependent material properties. If there are no temperature dependent material properties for a non-Piecewise Linear Analysis problem, PREMAT is wrapped up and a RETURN to the calling routine is executed.

For a non-Piecewise Linear Analysis problem for which a temperature set for material properties was selected in the user's Case Control Deck, all MATT1, MATT2 and MATT3 cards are read into open core from the MPT data block. For a Piecewise Linear Analysis problem MATS1 cards are read into open core from the MPT. A sorted list, with duplicates discarded, of the table numbers referenced on these cards is constructed in open core. This table number list is constructed so that every referenced table has eleven locations allocated to it. These eleven locations are used as a dictionary for the tables. The contents are: the table number (word 1); the table type 1,2,3, or 4 (word 2); pointers to the first and last entries in the table (words 3 and 4); parameters from the TABLE card (words 5 through 11). The DIT data block is then read. For each table read, it is determined by scanning the table number list whether or not the table is required for problem

solution. If it is required, the table is read into open core and the dictionary entry for the table is completed. For a required table which is a type 4 (polynomial) table, the functional values of the polynomial at the end points of the interval of the real line over which the polynomial is defined are calculated by an "internal subroutine" and stored in the table dictionary. If the table is not required, it is read until an end-of-table indicator is sensed. This process continues until all tables of the set TABLEM1, TABLEM2, TABLEM3 and TABLEM4 or of the set TABLES1, are exhausted. When all referenced tables have been read into core, PREMAT is wrapped up and a return is generated.

2. MAT: The basic logic of the MAT routine is straightforward. Eight types of table look-ups, described above for INFLAG = 1, 2, 3, 4, 5, 6, 7 and 8 are supported. A computed-go-to on INFLAG is executed and each option is carried out as described above. "Internal subroutines" which are entered via FØRTRAN ASSIGN and GØ TØ statements and return to their correct "calling" locations via ASSIGNED GØ TØ's are used liberally by MAT. It should be noted that each time MAT is called, MATID, INFLAG and other applicable input items, are saved. On the next call if the input is identical with the input of the previous call, nothing is stored in /MATØUT/. Hence, the calling routine should use /MATØUT/ as a "read-only data set".

3.4.36.5 Design Requirements

Subroutine GMMATS is the only non-root segment subroutine used by this routine. There are no other special requirements.

3.4.35.6 Diagnostic Messages

The following messages can be output via PREMAT and/or MAT: 3008, 2017, 2018, 2019, 2041, 2042, 2103, 2112, 2113, 2114, 2115, 2116, and 2117.

3.4.37 PRETRD (Utility for Modules Which Use the CSTM Data Block - Double Precision Version).

3.4.37.1 Entry Points: PRETRD, TRANSD

3.4.37.2 Purpose

A utility routine for modules which use the CSTM (Coordinate System Transformation Matrices) data block, TRANSD generates a real double precision 3x3 direction cosine matrix which maps a vector from a local coordinate system to basic coordinates. PRETRD sets up eventual calls to TRANSD. For a module to use TRANSD a call to PRETRD is made once and only once.

3.4.37.3 Calling Sequence

CALL PRETRD(CSTM,NCSTM)

CSTM = array of coordinate system transformation matrices (see data block description for CSTM, section 2.3) - mixed - input.

NCSTM = length of the CSTM array. NCSTM = 14*the number of coordinate systems in the CSTM data block - integer - input.

CALL TRANSD(ECPT,TA)

ECPT = array of length 4. The first word is an integer coordinate system identification number and the next 3 words are the components of a vector in basic coordinates - input only.

TA = real double precision 3x3 direction cosine matrix which maps a vector from the local coordinate system designated by ECPT(1) to basic coordinates - output.

3.4.37.4 Method

The CSTM array is searched to find a coordinate system transformation identification number that matches ECPT(1). If the coordinate system is rectangular, the 3x3 matrix, call it T, which is in words 6 through 14 of the CSTM blocks, is stored in TA and a RETURN is generated. If the coordinate system is basic, the identity matrix is returned. If the coordinate system is spherical or cylindrical, we calculate

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = [T]^T \begin{Bmatrix} E - V \end{Bmatrix}$$

where E is the input vector stored at ECPT(2) and V is the translation offset vector in basic

coordinates found in the CSTM block in words 3, 4 and 5; and

$$r = \sqrt{x^2 + y^2}$$

If the coordinate system is cylindrical define:

$$\left[T_\ell\right] = \begin{bmatrix} x/r & -y/r & 0 \\ y/r & x/r & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If the coordinate system is spherical define:

$$\ell = \sqrt{x^2 + y^2 + z^2},$$

$$\left[T_\ell\right] = \begin{bmatrix} x/\ell & \frac{xz}{r\ell} & -y/r \\ y/\ell & \frac{yz}{r\ell} & x/r \\ z/\ell & -r\ell & 0.0 \end{bmatrix}$$

Then TA = TT_ℓ is computed and the subroutine returns to the calling program.

3.4.37.5 Design Requirements

The routine is designed so that a module which uses the CSTM data block can have a utility routine to fetch a coordinate system transformation matrix. Typically, a module driver will attempt to open the file which contains the CSTM data block. If the data block is not purged, the module will read the entire data block into open core, close the file and call PRETRD to transmit the address of the array and the length of the array. Once this initialization call has been made, TRANSD may be called in the module as many times as necessary. The routine does not perform any I/Ø operations. The routine assumes the format of the CSTM data block, as outlined in the Data Block Description for the CSTM (section 2.3 of the Programmer's Manual) is correct, and no numerical checks are made.

3.4.37.6 Diagnostic Messages

If the coordinate system identification number transmitted via ECPT(1) can not be found in the CSTM array user fatal message 2025 occurs. The user should check coordinate system numbers on GRID bulk data cards against those defined on CØRD1C, CØRD1R, etc., bulk data cards to insure that there are no undefined coordinate systems.

3.4.38 PRETRS (Utility for Modules Which Use the CSTM Data Block - Single Precision Version).

3.4.38.1 Entry Points: PRETRS, TRANSS.

3.4.38.2 Purpose

A utility routine for modules which use the CSTM (Coordinate System Transformation Matrices) data block, TRANSS generates a real single precision 3x3 direction cosine matrix which maps a vector from a local coordinate system to basic coordinates. PRETRS sets up eventual calls to TRANSS. For a module to use TRANSS a call to PRETRS is made once and only once.

3.4.38.3 Calling Sequence

```
CALL PRETRS(CSTM,NCSTM)

CALL TRANSS(ECPT,TA)
```

This routine is exactly the same as subroutine PRETRD (see section 3.4.37) and TRANSD except that TRANSD, an entry point, returns a real double precision matrix TA and uses double precision arithmetic, while TRANSS returns a real single precision matrix TA and uses single precision arithmetic. See subroutine description for PRETRD for details on subroutine arguments, method, design requirements and diagnostic messages.

3.4.39 PRETAB (Table Look-Up).

3.4.39.1 Entry Points: PRETAB, TAB.

3.4.39.2 Purpose

To read tables (functions) from the data block DIT, Direct Input Tables, into core and to set up table dictionaries which are subsequently used when the calling routine requests a functional value from a table via a call to the entry point TAB. The routine is designed so that PRETAB is called once and only once by a module and so that TAB may be called many times as a table look-up routine.

3.4.39.3 Calling Sequence

CALL PRETAB(DITF,Z,IZ,BUF,LCRGVN,LCUSED,TABNØL,LIST)

DITF - GINØ file number of the Direct Input Tables data block - integer - input.

Z - Array of core given to the subroutine as working storage - real - input and output.

IZ - Same address as Z. Used as integer in this routine.

BUF - A GINØ buffer (plus one cell) used by subroutine PRELØC - input.

LCRGVN - The length of Z array, given to PRETAB and TAB - integer - input.

LCUSED - The number of cells of core used by PRETAB - integer - output.

TABNØL - List of table numbers that the calling routine will be referencing via TAB calls. TABNØL(1) = N is the number of tables to be referenced. TABNØL(2), ... , TABNØL(N +1) contain the table numbers. Note that 0 is an admissible table number. Table 0 defines a function which is identically zero for all values of the independent variable - integer - input.

LIST - Array of control words for subroutine LØCATE and table types. LIST(1) = M is the number of triples which follow in the list. The first two words of each triple are the subroutine LØCATE control words for the particular table being referenced and the third word is the table type: 1, 2, 3 or 4 - integer - input.

CALL TAB(TABID,X,Y)

TABID - Table number - integer - input.

X - Abscissa for table number TABID at which the functional value is desired - real - input.

Y - Functional value (ordinate) of abscissa X for table number TABID - real - output.

3.4.39.4 Method

PRETAB: For each table in the TABNØL list an 11 word table dictionary entry is defined in open core. The first word in each entry is the table number obtained from the TABNØL list. Then the DIT data block is read. For each entry of the DIT, it is determined whether or not this table number is in the TABNØL list. If it is not, then the table is read serially until an end-of-table indicator is sensed. If it is a table called for in the TABNØL list, the program sets words 2 and 3 of the table dictionary, the table type (1,2,3 or 4) and the pointer to the lst entry in the table respectively. The table is then read into core, and the 4th word of the table dictionary, the pointer to the last entry in the table, is set. Words 5 through 11 of the dictionary, the table parameters, are set. If the table type is 4, indicating a polynomial, the functional values of the polynomial at the end points of the interval of the real line over which the polynomial is defined are calculated. After the tables for an entry in the LIST array have been exhausted, a check is made to determine if all tables in the TABNØL list have been found (after each table is found the table number is set negative). If all tables have been found, the table numbers in TABNØL are set to their original positive status and the routine is wrapped up. If all tables have not been found, the next class of table cards, defined by the next triple in the LIST array, are located in the DIT data block and the process is repeated.

TAB: The table dictionary is searched until a match is found with the input argument TABID. The table type (1, 2, 3 or 4) is determined, the (functional) argument is computed after a 4-way branch on table type, and a transfer is made to either the "internal subroutine" which performs linear interpolation--if the table type is 1, 2 or 3--or the "internal subroutine" which performs polynomial evaluation--if the table type is 4.

3.4.39.5 Design Requirements

DIT must not be purged. Enough open core must be made available to construct the table dictionaries and to contain all referenced tables in core. All table numbers must be unique. All table numbers input via the TABNØL array must be found in the DIT data block. A table number referenced by the TABID argument of TAB must have been referenced previously in the TABNØL array.

3.4.39.6 Diagnostic Messages

The following diagnostic messages may appear:

3008

2088

2089

2090

3.4.40 AXIS (Draw an Axis on a Plot).

3.4.40.1 Entry Point: AXIS.

3.4.40.2 Purpose

To draw an x or y axis on a plotter.

3.4.40.3 Calling Sequence

CALL AXIS(X1,Y1,X2,Y2,PEN,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X1,YL = starting point of the axis line - real - input.

X2,Y2 = terminal point of the axis line - real - input.

PEN = pen number or line density to be used (its meaning depends on the plotter) - integer - input.

ØPT = { -1 to initiate the line mode. / +1 to terminate a series of plot commands. / 0 to draw an axis. } - integer - input

/PLTDAT/

MØDEL = plotter model number - integer - input.

PLØTER = plotter number (i) - integer - input.

NPENS = largest number of pens or maximum density for plotter i - integer - input.

3.4.40.4 Method

This subroutine calls LINE or AXISi, depending on whether the plotter has available a single command used for drawing an axis. At this writing, only plotter 3 has a special axis command.

If ØPT $\neq$ 0, all other arguments are ignored, and LINE or AXISi is called. Otherwise, alternate pen number (PENX) is calculated modulo NPENS and is used as the pen number passed to

LINE or AXISi, as follows:

```
PENX = PEN - NPENS * ((PEN-1)/NPENS)
```

3.4.40.5 Design Requirements

Generally, AXIS or LINE should be called with ØPT = -1 before axes are generated, even though it is not necessary to specifically put all plotters in the line mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the typing mode).

Subroutines used: LINE, AXISi.

3.4.41 AXISi (Axis Routine for Plotter i).

3.4.41.1 Entry Point: AXISi.

3.4.41.2 Purpose

To set up a plot command to draw an x or y axis on plotter i.

3.4.41.3 Calling Sequence

CALL AXISi(X1,Y1,X2,Y2,PEN,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X1,Y1 = starting point of the axis line - real - input.

X2,Y2 = terminal point of the axis line - real - input.

PEN = pen number or line density to be used (meaning depends on plotter) - integer - input.

ØPT = { -1 to initiate the line mode
+1 to terminate a series of plot commands
0 to draw an axis } - integer - input.

/PLTDAT/

XYMIN = minimum x and y values of the region permitted on plotter i - real - input.

XYMAX = maximum x and y values of the region permitted on plotter i - real - input.

ØRIGIN = location of the lower left corner of the plotter relative to its true physical origin - real - input.

3.4.41.4 Method

Taking into account the true origin of the plotter, the plot command is generated.

3.4.41.5 Design Requirements

Subroutine used: WPLTi.

3.4.42 SKPFRM (Skip a Variable Number of Frames).

3.4.42.1 Entry Point: SKPFRM.

3.4.42.2 Purpose

Tc skip a variable number of frames, if appropriate to the plotter.

3.4.42.3 Calling Sequence

CALL SKPFRM (BFRAMS)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

BFRAMS = number of frames to be skipped - integer - input.

/PLTDAT/

MØDEL = plotter model number - integer - input.

PLØTER = plotter number - integer - input.

REG = plot region parameters - real - input.

AXYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - input.

EDGE = size of the borders (x,y) in plotter units - real - input.

CAMERA = currently active camera - integer - input.

ØRIGIN = location (x,y) of the lower left corner of the plotter relative to its true physical origin - real - input.

3.4.42.4 Method

For plotters 3 and 9, the specified number of frames (BFRAMS) are skipped. For plotters 4 to 7, the remainder of the current plot is skipped, and another half plot is also skipped. For plotters 1, 2 and 4, nothing is done due to the absence of any automatic method of skipping blank paper.

3.4.42.5 Design Requirements

Subroutines used: LINE, WPLTi.

3.4.43 SELCAM (To Initiate a New Plot).

3.4.43.1 Entry Point: SELCAM.

3.4.43.2 Purpose

To select a camera and/or to generate a setup record for a new plot.

3.4.43.3 Calling Sequence

CALL SELCAM (CAMERA,PLTNUM,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

CAMERA = camera number to be selected (if appropriate) - integer - input.

PLTNUM = plot number - integer - input.

ØPT ≠ 0 = if the camera is to be selected when appropriate, and nothing is to be done when not appropriate - integer - input.

/PLTDAT/

MØDEL = plotter model number - integer - input.

PLØTER = plotter number - integer - input.

XYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - input.

EDGE = size of the borders (x,y) in plotter units - real - input.

CAMNUM = last selected camera - integer - output.

ØRIGIN = location (x,y) of the lower left corner of the plotter relative to its true physical origin - real - input and output.

3.4.43.4 Method

If ØPT ≠ 0 and a camera is not appropriate to the plotter, nothing is done by this subroutine.

Otherwise, what is done is dependent upon the plotter hardware requirements.

For plotters 1, 2 and 8, the plotter is stopped with the plot number displayed in the console lights. For plotters 2 and 9 the specified camera is selected. And for plotters 4 to 7, a block address record with the plot number is generated.

3.4.43.5 Design Requirements

Subroutines used: WPLTi, LINE.

3.4.44 IDPLØT (Generate an "ID" Plot).

3.4.44.1 Entry Point: IDPLØT.

3.4.44.2 Purpose

To identify the owner of all the plots by printing the information contained on the PLØTID card in the user's Case Control Deck prior to generating the first plot.

3.4.44.3 Calling Sequence

```
CALL IDPLØT (IDX)

CØMMØN/ØUTPUT/SKIP(32,6),ID(32)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.
```

where:

IDX = { 0 if a plot id was not generated / 1 if a plot id was generated } - integer - output.

/ØUTPUT/

ID = user supplied PLØTID, in the Case Control Deck - BCD - input.

/PLTDAT/

XYMIN, XYMAX = { plot region parameters - real - input.

AXYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - input.

EDGE = size of the borders (x,y) in plotter units - real - input.

CNTX,CNTY= number of counts per printed character in the x and y directions respectively - real - input.

PLTYPE = plotter type - integer - input.

3.4.44.4 Method

If there is no PLØTID (ID = blanks), IDX is set to zero and no identification is generated. Otherwise, IDX is set to one and an identification is generated. The current region parameters are saved (they will be restored at the end of the subroutine) and are set to include the entire paper area. The identification generated varies, depending upon the plotter type.

If the plotter is a microfilm plotter (|PLTYPE| = 1), an entire frame is generated as identification. The top and bottom of the frame are a series of closely spaced horizontal lines. The PLØTID is then printed three times in the center of the frame.

If the plotter is a drum or table plotter (|PLTYPE| ≠ 1), the identification is printed once at the very bottom of the paper within the bottom border.

After the identification is generated, the PLØTID is set to blanks. This insures that the identification will be generated prior to the first plot only.

3.4.44.5 Design Requirements

Subroutines used: AXIS, PRINT.

3.4.45 INTGPX (Search a List of Integers).

3.4.45.1 Entry Points: INTGPX, INTGPT.

3.4.45.2 Purpose

Given a list of N integers, to find the index of the list item equal to ITEM (primarily used to search a list of external grid point id's).

3.4.45.3 Calling Sequence

```
CALL INTGPX(LIST,N)
K = INTGPT (ITEM)
```

where:

LIST = list of N integers, in arbitrary order - input.

N = number of entries in LIST - integer - input.

ITEM = integer for which a match is to be found in LIST - input.

3.4.45.4 Method

Search LIST using a linear search until a match for ITEM is found. Then the result (INTGPT) is set equal to the index of LIST where the match occurs. If no match is found, the result is set = 0.

3.4.45.5 Design Requirements

INTGPX must be called before INTGPT is used. As long as LIST does not change location and the value of N does not change, INTGPX need not be called again.

3.4.46 INTLST (Interpret a List of Integers).

3.4.46.1 Entry Point: INTLST.

3.4.46.2 Purpose

To interpret a list of integers and/or pairs of integers separated by the word TØ or THRU.

3.4.46.3 Calling Sequence

CALL INTLST(LIST,N,SIGN,N1,N2)

where:

LIST - the list to be interpreted - integer - input.

N - index location of the next list item(s) to be interpreted - integer - input.

SIGN - sign (±1) of the interpreted integer or the first of a pair of integers - output.

N1 - absolute value of the interpreted integer or the first of a pair of integers - output.

N2 - absolute value of the second integer of pair of integers (= N1 if not a pair) - output.

3.4.46.4 Method

SIGN = ±1 if LIST(N) is positive or negative.

N1 = absolute value of LIST(N).

If LIST(N+1) ≠ TØ or THRU, then N2 = N1 and N is incremented by 1.

If LIST(N+1) = TØ or THRU, then N2 = absolute value of LIST(N+2) and N is incremented by 3.

3.4.46.5 Design Requirements

Initially, N must be set to the index of the first integer or integer pair to be interpreted in LIST. If the list is consecutive, N need not subsequently be altered until a new list is to be interpreted. It is advisable that the value following the last item in LIST be set = 0 to avoid the chance that it may equal TØ or THRU.

3.4.47 LINE (Draw a Line on a Plotter).

3.4.47.1 Entry Point: LINE.

3.4.47.2 Purpose

To draw a line on a plotter.

3.4.47.3 Calling Sequence

CALL LINE(X1,Y1,X2,Y2,PEN,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X1,Y1 = starting point of the line - real - input.

X2,Y2 = terminal point of the line - real - input.

PEN = pen number or line density to be used - integer - input.

ØPT = { -1 to initiate the line mode.
+1 to terminate a series of plot commands.
0 to draw a line. } integer - input.

/PLTDAT/

MØDEL = plotter model number - integer - input.

PLØTER = plotter number (i) - integer - input.

REG = x and y values defining the region in which the line is to be drawn - real - input.

NPENS = maximum number of pens or line density possible for plotter i - integer - input.

3.4.47.4 Method

If the line to be drawn is entirely outside the specified region, the subroutine immediately returns without drawing anything. If only part of the line is outside the region, only that portion of the line within the region is drawn. The actual pen number or line density used will be modulo the maximum number of pens or line density as follows:

PENX = PEN - NPENS* ((PEN-1)/NPENS)

Then LINEi is called.

3.4.47.5 Design Requirements

Generally, LINE should be called with ØPT = -1 before any lines are drawn, even though it is not necessary to specifically put all plotters in the line mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the typing mode). If ØPT ≠ 0, all other arguments are ignored. Subroutine used: LINEi.

3.4.48 LINEi (Draw a Line on Plotter i).

3.4.48.1 Entry Point: LINEi.

3.4.48.2 Purpose

To draw a line on plotter i.

3.4.48.3 Calling Sequence

CALL LINEi(X1,Y1,X2,Y2,PEN,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description section 2.5.

where:

X1,Y1 = starting point of the line - real - input.

X2,Y2 = terminal point of the line - real - input.

PEN = pen number or line density to be used - integer - input.

ØPT = { -1 to initiate the line mode.
+1 to terminate a series of plot commands.
 0 to draw a line. } - integer - input.

/PLTDAT/

MØDEL = plotter model number - integer - input.

PLØTER = plotter number - integer - input.

MAXLEN = maximum length of a line segment - real - input.

ØRIGIN = x and y values of the current position of the pen (applicable only to incremental plotters) - real - input and output.

3.4.48.4 Method

If ØPT ≠ 0, all other arguments are ignored. If ØPT = -1 and if applicable for plotter i, a flag is set so that when LINEi is subsequently called with ØPT = 0, the plotter will be put into the line mode before drawing the requested line. If ØPT = +1 and if applicable for plotter i, the pen is raised. Then, no matter which plotter is being used the current sequence of plotter commands is terminated. If ØPT = 0, the line is drawn as a series of line segments, each of maximum length MAXLEN.

3.4.48.6 Design Requirements

Subroutines used: WPLTi.

3.4.49 PRINT (Print a Title on a Plotter).

3.4.49.1 Entry Point: PRINT.

3.4.49.2 Purpose

To type a title on a plotter horizontally or vertically.

3.4.49.3 Calling Sequence

CALL PRINT(X,Y,XYD,CHR,N,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X,Y - starting or ending point of the title to be typed (always left-to-right or top-to-bottom) - real - input.

XYD - $\begin{cases} \pm 1 \text{ if X = starting or ending point of the title - integer - input.} \\ \pm 2 \text{ if Y = starting or ending point of the title - integer - input.} \end{cases}$

CHR - title to be typed (four characters/word - left adjusted followed by blanks) - BCD - input.

N - number of words in the title - integer - input.

ØPT - $\left\{ \begin{array}{l} -1 \text{ to initiate the typing mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to type a title.} \end{array} \right\}$ - integer - input.

/PLTDAT/

CNTCHR = number of plotter counts per character in the x and y directions - real - input.

3.4.49.4 Method

If ØPT ≠ 0, all other arguments are ignored and TIPE is called. Otherwise, each character in the title (CHR) is separated and put into another array (C). This is done for each 20 words of the title (80 characters), and TIPE is then called to type these characters.

3.4.49.5 Design Requirements

Generally, one of the typing subroutines (PRINT, TIPE, TYPFLT, TYPINT, SYMBØL) should be

called with ØPT = -1 before any typing is attempted, even though it is not necessary to specifically put all plotters in the typing mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the line mode).

Subroutines used: TIPE.

3.4.50 RDMØDX (Read a File Containing XRCARD Translations).

3.4.50.1 Entry Points: RDMØDX, RDMØDY, RDMØDE, RDWØRD.

3.4.50.2 Purpose

Tc read from a file or storage a record containing the subroutine XRCARD interpretation of free field data cards (e.g., the PCDB data block).

3.4.50.3 Calling Sequence

```
CALL RDMØDX(FILE,MØDE,WØRD)
CALL RDMØDY(A,MØDE,WØRD)
CALL RDMØDE($n1,$n2,$n3)
CALL RDWØRD
```

where:

FILE = GINØ file name which is to be read - integer - input.

MØDE = storage location into which the XRCARD mode value is to be read - integer - output.

WØRD = 2 locations into which XRCARD card items are to be read - integer - output.

A = array which is to be "read" (instead of FILE) - integer - input.

n_1 = the FØRTRAN statement number defining the return at which numeric data are interpreted (MØDE < 0).

n_2 = the FØRTRAN statement number defining the return at which alphabetic data are interpreted (0 < MØDE < 1,000,000).

n_3 = the FØRTRAN statement number defining the return when the end of a logical card is encountered (MØDE $\geq$ 1,000,000).

3.4.50.4 Method

RDMØDX and RDMØDY are initialization calls for the file and core oriented options respectively.

For RDMØDE:

1. An XRCARD mode value is read into MØDE. If MØDE = 0, the end of a physical card has been encountered, but not the end of a logical card. In this case, the record is terminated (if FILE is being read). Then the first word of the next record or location is read into MØDE.

2. If MØDE < 0, the next word is read into WØRD(1). If MØDE = -4, another word is read into WØRD(2).

3. If 0 < MØDE < 1,000,000, BCD information follows as pairs of 4-character words. The first two of these words are read into NEXT(1) and NEXT(2). If NEXT(1) is a blank or NEXT(2) is a delimiter, the value of MØDE is decremented by one, and if MØDE is still greater than zero, the next two words are read into NEXT(1) and NEXT(2). This continues until either MØDE = 0, or NEXT(1) is not a blank and NEXT(2) is not a delimiter. If MØDE does become zero, step 1 is then re-executed.

4. If MØDE $\geq$ 1,000,000, the end of a logical card has been encountered. If FILE is being read, the current record is terminated.

For RDWØRD:

1. The two words now in NEXT(1) and NEXT(2) are stored in WØRD(1) and WØRD(2).

2. MØDE is decremented by one. If MØDE is still gerater than zero, the next two words are read into NEXT(1) and NEXT(2). If NEXT(1) is a blank or NEXT(2) is a delimiter, this step is repeated until either MØDE = 0, or NEXT(1) is not a blank and NEXT(2) is not a delimiter.

3.4.50.5 Design Requirements

RDMØDX or RDMØDY must be called before RDMØDE and RDWØRD. As long as FILE does not change in value, and MØDE, WØRD, and A do not change locations, RDMØDX and RDMØDY need not be called again. If RDMØDX is called, FILE must be opened and properly positioned by the calling program. In addition, RDMØDE and RDWØRD cannot be called when FILE is closed. If an end-of-file or-record condition is encountered, a fatal error occurs (see subroutine FREAD).

3.4.51 SGINØ (GINØ for Unformatted Tapes).

3.4.51.1 Entry Points: SØPEN, SWRITE, SEØF, SCLØSE.

3.4.51.2 Purpose

To write unformatted BCD and binary tapes to drive NASTRAN plotters.

3.4.51.3 Calling Sequences

CALL SØPEN($n,PLTTP,BUFFER,LBUFF)

n -- FØRTRAN statement number defining the return if PLTTP is not available for writing.

PLTTP - GINØ file name of the plot tape. This may have two values: PLT1 - BCD plot tape; PLT2 - binary plot tape - BCD - input.

BUFFER - Array in which the plot data transmitted during SWRITE calls are stored.

LBUFF - Length of the buffer array - integer - input.

CALL SWRITE(PLTTP,DATA,LDATA,IØPT)

PLTTP - GINØ file name of the plot tape - BCD - input.

DATA - Array of plot data (1 character/word, right justified, leading zeros).

LDATA - Length of the DATA array in words - integer - input.

IØPT - { 0, potentially more data to be transmitted in this record. / 1, end of record with this data transmission. } integer - input.

CALL SEØF(PLTTP)

PLTTP - GINØ file name of the plot tape on which a physical EØF will be written.

CALL SCLØSE(PLTTP)

PLTTP - GINØ file name of the plot tape.

3.4.51.4 Method

SGINØ stores data in BUFFER until IØPT = 1 or BUFFER is filled. It then transmits the data to a physical tape without any control words. The data are transmitted to SGINØ 1 character (right

justified, leading zeros) per word. SGINØ packs these characters into full words. SGINØ is in FØRTRAN on all machines. On the IBM 7094 it interfaces with GINØ; the Univac 1108 version uses NTRAN; the IBM S/360 uses FØRTRAN I/Ø; and the CDC 6600 use XIØRTNS. See section 5 for details.

3.4.51.5 Design Requirements

Only one of PLT1 or PLT2 may be open at one time.

SØPEN must be called before SWRITE, SEØF, or SCLØSE.

PLT1 or PLT2 must be physical tapes if they are written on.

3.4.52 STPLØT (To Initiate a New Plot or Terminate the Current Plot).

3.4.52.1 Entry Point: STPLØT.

3.4.52.2 Purpose

To initiate a new plot or terminate the current plot.

3.4.52.3 Calling Sequence

CALL STPLØT(PLTNUM)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

CØMMØN/XXPARM/ - see XXPARM Miscellaneous Table description, section 2.5.

where:

PLTNUM = { if nonnegative, the plot number / if negative, terminate the current plot } - integer - input.

/PLTDAT/

MØDEL = plotter model index - integer - input.

PLØTER = plotter number - integer - input.

REG = plot region parameters - real - input.

XYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - input.

PLTYPE = plotter type - integer - input.

PLTAPE = plot tape - BCD - input.

EØF = { 0 if an end-of-file mark is to be written on the plot tape after each plot / 1 if no end-of-file mark is to be written on the plot tape after each plot } - integer - input.

/XXPARM/

CAMERA = camera number (if applicable) to be used - integer - input.

BFRAMS = number of blank frames between plots - integer - input.

3.4.52.4 Method

A. If PLTNUM is nonnegative:

1. Select the specified camera or create a setup record appropriate to the plotter (CALL SELCAM).

2. Skip to a new frame (if applicable) and create the owner identification. If the owner identification is generated by subroutine IDPLØT, re-execute step 1 and skip to a new frame.

3. If appropriate to this plotter, insert the desired number of blank frames on film only. If the camera specified is camera 2 (paper only), no blank frames are inserted.

4. If the plot number is nonzero, type this number in the upper left and right corners of the picture.

B. If PLTNUM is negative:

1. Terminate the current plot tape record.

2. Close the current plot tape file (CALL SCLØSE).

3. If each plot is to be separated by an end-of-file mark (EØF = 0), write an end-of-file on the plot tape (CALL SEØF).

3.4.52.5 Design Requirements

Subroutines used include: IDPLØT, SELCAM, SKPFRM, TYPINT, SCLØSE, SEØF.

3.4.53 SYMBØL (Type a Symbol on a Plotter).

3.4.53.1 Entry Point: SYMBØL.

3.4.53.2 Purpose

To type a symbol on a plotter.

3.4.53.3 Calling Sequence

CALL SYMBØL(X,Y,SYM,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

CØMMØN/SYMBLS/ - see SYMBLS Miscellaneous Table description, section 2.5.

where:

X,Y - point at which the symbol is to be typed - real - input.

SYM - two consecutive storage locations each containing an index into the SYMBLS table - integer - input.

ØPT - { -1 to initiate the typing mode. / +1 to terminate a series of plot commands. / 0 to type the symbol. } - integer - input.

/PLTDAT/

MØDEL - plotter model number - integer - input.

PLØTER - plotter number (i) - integer - input.

/SYMBLS/

NSYM - number of symbols defined in the SYMBLS table - integer - input.

SYMBL(20,i) - character indices defining the symbols of plotter i - integer - input.

3.4.53.4 Method

If ØPT $\neq$ 0, all other arguments are ignored and TYPEi or DRWCHR is called. Otherwise, an alternate symbol index (SYMX) is calculated modulo NSYM for each index in SYM and is used as the actual symbol index, as follows:

$$SYMX = SYM_j - NSYM*((SYM_{j-1})/NSYM) \quad , \quad j = 1, 2.$$

Then TYPEi or DRWCHR is called for each symbol.

The reason for SYM being two indices is to enable the user to create any additional symbol by combining any two of the valid symbols in the SYMBLS table. Note: any of the indices in SYM may = 0. This would imply that a new symbol is not being created.

3.4.53.5 Design Requirements

Generally, one of the typing subroutines (PRINT, TIPE, TYPFLT, TYPINT, SYMBØL) should be called with ØPT ≠ -1 before any typing is attempted, even though it is not necessary to put all plotters in the typing mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the line mode).

Subroutines used: TYPEi, DRWCHR.

3.4.54 TIPE (Type a Line of Characters on a Plotter).

3.4.54.1 Entry Point: TIPE.

3.4.54.2 Purpose

To type a line of characters on a plotter horizontally or vertically.

3.4.54.3 Calling Sequences

CALL TIPE (X,Y,XYD,CHR,N,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

CØMMØN/CHAR94/ - see CHAR94 Miscellaneous Table description, section 2.5.

where:

X,Y - starting or ending point of the line to be typed (always left-to-right or top-to-bottom) - real - input.

XYD - $\left\{\begin{array}{l}\pm 1 \text{ if X = starting or ending point of the line.} \\ \pm 2 \text{ if Y = starting or ending point of the line.}\end{array}\right\}$ - integer - input.

CHR - line of characters to be typed (one character/word - left adjusted followed by blanks) - BCD - input.

N - number of characters to be typed - integer - input.

ØPT - $\left\{\begin{array}{l}-1 \text{ to initiate typing mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to type a line of characters.}\end{array}\right\}$ - integer - input.

/PLTDAT/

MØDEL - plotter model number - integer - input.

PLØTER - plotter number (i) - integer - input.

CNTCHR - number of plotter counts per character in the x and y directions - real - input.

/CHAR94/

CHAR - Section I of the CHAR94 table - BCD - input.

3.4.54.4 Method

If ØPT ≠ 0, all other arguments are ignored and TYPEi or DRWCHR is called. Otherwise, for each character to be typed, an index into the CHAR character set is found. This is done 80 characters at a time. If a character cannot be located, it is treated as a blank. For each set of 80 character indices set up, TYPEi or DRWCHR is called.

3.4.54.5 Design Requirements

Generally, one of the typing subroutines (PRINT, TIPE, TYPFLT, TYPINT, SYMBØL) should be called with ØPT ≠ -1 before any typing is attempted, even though it is not necessary to put all plotters in the typing mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the line mode).

Subroutines used: TYPEi, DRWCHR.

3.4.55 TYPEi (Type a Line of Characters on Plotter i).

3.4.55.1 Entry Point: TYPEi.

3.4.55.2 Purpose

To type a line of characters on plotter i horiziontally or vertically.

3.4.55.3 Calling Sequence

CALL TYPEi(X,Y,XYD,CHR,N,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

CØMMØN/CHAR94/ - see CHAR94 Miscellaneous Table description, section 2.5.

where:

X,Y = starting or ending point of the line to be typed (always right-to-left or top-to bottom) - real - input.

XYD = { ±1 if x = starting or ending point of the line. / ±2 if y = starting or ending point of the line. } - integer - input.

CHR = indices of the line of characters to be typed (see description for TIPE, section 3.4.54) - integer - input.

N = number of characters to be typed - integer - input.

ØPT = { -1 to initiate the typing mode. / +1 to terminate a series of plot commands / 0 to type a line of characters. } - integer - input.

/PLTDAT/

XYMIN = minimum x and y values of the region in which the line is to be typed - real - input.

XYMAX = maximum x and y values of the region in which the line is to be typed - real - input.

CNTCHR = number of plotter counts per character in the x and y directions - real - input.

/CHAR94/

CHRCØD = Section II, III, or IV of the CHAR94 table - integer - input.

3.4.55.4 Method

If ØPT ≠ 0, all other arguments are ignored. If ØPT = -1 and if applicable for plotter i, a flag is set so that when TYPEi is subsequently called with ØPT = 0, the plotter will be put into the typing mode before typing the first character. If ØPT = +1, the current sequence of plotter commands is terminated.

Define:

LSTCHR = last legitmate character index for plotter i.

NCHR = number of character indices which must be changed for plotter i.

CHAR = NCHR pairs of character indices. The first index of each pair is the index which must be changed, and the second index is the replacement index.

If $N \leq 0$, it is assumed that one character is to be typed.

Each character index in CHR is checked against LSTCHR. If the index is greater than LSTCHR, a blank is inserted at the corresponding point on the plot. Otherwise, indices are altered if need be from CHAR and the character is typed.

No characters will be typed outside the region as defined by XYMIN and XYMAX.

3.4.55.5 Design Requirements

Subroutines used: WPLTi.

3.4.56 TYPFLT (Type a Floating Point Number on a Plotter).

3.4.56.1 Entry Point: TYPFLT.

3.4.56.2 Purpose

To type a floating point number on a plotter, horizontally or vertically.

3.4.56.3 Calling Sequence

CALL TYPFLT (X,Y,XYD,V,FIELD,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X,Y - point at which the number is to be typed (always left-to-right or top-to-bottom) - real - input.

XYD - $\left\{\begin{array}{l}\pm 1 \text{ if X = starting or ending point of the typed number.} \\ \pm 2 \text{ if Y = starting or ending point of the typed number.}\end{array}\right\}$ - integer - input.

V - number to be typed - real - input.

FIELD - field width of the typed number. If FIELD > 0, the number will be centered at (X,Y). If FIELD < 0, the number will be typed starting or ending at (X,Y). If |XYD| = 1 or 2, the number will be typed horizontally or vertically respectively - integer - input.

ØPT - $\left\{\begin{array}{l}-1 \text{ to initiate the typing mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to type the number.}\end{array}\right\}$ - integer - input.

/PLTDAT/

MØDEL - plotter model number - integer - input.

PLØTER - plotter number (i) - integer - input.

CNTCHR - number of plotter counts per character in the x and y directions - real - input.

3.4.56.4 Method

If ØPT ≠ 0, all other arguments are ignored and TYPEi or DRWCHR is called. Otherwise, the

number of significant digits (NSIG) to be typed is determined.

If $V \geq 0$, the typed number will be unsigned. If FIELD > 4, the number of significant digits typed will be at least = FIELD - 4. If FIELD $\leq$ 4, NSIG = FIELD - 1.

If $V < 0$, the typed number will be signed. If FIELD > 5, the number of significant digits typed will be at least FIELD - 5. If FIELD $\leq$ 5, NSIG = FIELD - 2.

The number (V) is multiplied by some power of ten such that the product is between 10^7 and 10^8. It can then be expressed as an 8-significant digit integer. If the number is such that NSIG digits cannot be typed without an exponent, a standard form is used: -X.XXXX ... $\pm$ XX. Otherwise the decimal point is adjusted and the exponent will not be printed.

3.4.56.5 Design Requirements

Generally, one of the typing subroutines (PRINT, TIPE, TYPFLT, TYPINT, SYMBØL) should be called with ØPT = -1 before any typing is attempted, even though it is not necessary to put all the plotters in the typing mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the line mode).

Subroutines used: TYPEi, DRWCHR.

3.4.56.6 Diagnostic Messages

If NSIG significant digits cannot possibly be typed in the field width (FIELD) specified, the entire field will be filled with asterisks (**...*).

3.4.57 TYPINT (Type an Integer Number on a Plotter).

3.4.57.1 Entry Point: TYPINT.

3.4.57.2 Purpose

To type an integer number on a plotter, horizontally or vertically.

3.4.57.3 Calling Sequence

CALL TYPINT (X,Y,XYD,NUM,FIELD,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X,Y - point at which the number is to be typed (always left-to-right or top-to-bottom) - real - input.

XYD - { ±1 if X = starting or ending point of the typed number. / ±2 if Y = starting or ending point of the typed number. } - integer - input.

NUM - number to be typed - integer - input.

FIELD - { +1 if the typed number is to be centered at (X,Y). If |XYD| = 1 or 2, the number will be typed horizontally or vertically, respectively. / -1 or 0, the number will be typed starting or ending at (X,Y). If FIELD = -1, FIELD will be set to the number of digits typed by the subroutine; in this case, FIELD must be a symbol in the call statement. - integer - input and output. }

ØPT - { -1 to initiate the typing mode. / +1 to terminate a series of plot commands. / 0 to type the number. } - integer - input.

/PLTDAT/

MØDEL - plotter model number - integer - input.

PLØTER - plotter number (i) - integer - input.

CNTCHR - number of plotter counts per character in the x and y directions - real - input.

3.4.57.4 Method

If ØPT ≠ 0, all other arguments are ignored and TYPEi or DRWCHR is called. Otherwise, each digit of the number is separated and used as character indices for the TYPEi or DRWCHR subroutines. In addition, if FIELD < 0, FIELD is set = the number of digits printed.

3.4.57.5 Design Requirements

Generally, one of the typing subroutines (PRINT, TIPE, TYPFLT, TYPINT, SYMBØL) should be called with ØPT = -1 before any typing is attempted, even though it is not necessary to specifically put all plotters in the typing mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the line mode).

Subroutines used: TYPEi, DRWCHR.

3.4.58 WPLT1 (Write a Plotter Command for Plotter 1).

3.4.58.1 Entry Point: WPLT1.

3.4.58.2 Purpose

To write a plotter command for plotter 1.

3.4.58.3 Calling Sequence

CALL WPLT1 (A,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

CØMMØN/CHAR94/ - see CHAR94 Miscellaneous Table description, section 2.5.

where:

A(1) - x-coordinate (integer).

A(2) - y-coordinate (integer).

A(3) - annotation character index (a).

A(4) - control character index (c_1).

A(5) - control character index (c_2).

A(6) - control character index (c_3).

ØPT - { 0, if A is a plot command
1, if a series of plot commands is to be terminated } integer - input.

/PLTDAT/

EDGE - size of the x and y borders of the paper - real - input.

PLØT - GINØ file name of the plot tape to be written - BCD - input.

/CHAR94/

CHAR(60,3) - sections II, III, and IV of the CHAR94 character table - integer - input.

3.4.58.4 Method

If computer 1 (IBM 7094) is the computer being used, section II of the CHAR94 table is used as the characters written on the plot tape; if computer 4 (CDC 6600) section IV is used; if any

other computer, section III is used.

The lower left corner of the paper is assumed to be at (0,0). Taking into account the x and y borders, the true x and y coordinates are calculated. These coordinates are then separated into four integer digits. The plot command is then set up and written as follows:

$$rbx_4x_3x_2x_1bby_4y_3y_2y_1bc_1c_2bbc_3babbbbb$$

preceeded by 35 blanks. The resulting plot command is 60 characters long.

r = record mark (character 49 in the CHAR94 table)

b = blank (character 48 in the CHAR94 table)

x_i = x-coordinate digit

y_i = y-coordinate digit

c_i = control character

a = annotation character.

3.4.58.5 Design Requirements

Subroutine used: SWRITE.

3.4.59 WPLT2 (Write a Plotter Command for Plotters 2 and 8)

3.4.59.1 Entry Point: WPLT2.

3.4.59.2 Purpose

To write a plot command for plotters 2 and 8.

3.4.59.3 Calling Sequence

CALL WPLT2 (A,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

CØMMØN/CHAR94/ - see CHAR94 Miscellaneous Table description, section 2.5.

where:

A(1) = x-coordinate (integer).

A(2) = y-coordinate (integer).

A(3) = annotation character index (a).

A(4) = control character index (c_1).

A(5) = control character index (c_2).

A(6) = control character index (c_3).

ØPT = {0, if A = plot command; 1, if a series of plot commands is to be terminated} - integer - input.

/PLTDAT/

AXYMAX - size of the paper (in plotter units) being used, less the border - real - input.

PLØT - GINØ file name of the plot tape to be written - BCD - input.

/CHAR94/

CHAR(60,3) - sections II, III and IV of the CHAR94 table - integer - input.

3.4.59.4 Method

If computer 1 (IBM 7094) is the computer being used, section II of the CHAR94 table is used for the characters written on the plot tape; if computer 4 (CDC 6600), section IV is used; if any

other computer, section III is used.

Assuming the true physical origin of the plotter to be at the center of the paper, the true x and y coordinates are calculated. These coordinates are then separated into four integer digits. A plot command is then set up and written as follows:

$$Xs_x x_4 x_3 x_2 x_1 Ys_y y_4 y_3 y_2 y_1 bbbbbbbbc_1 c_2 ac_3$$

preceeded by 96 blanks. The resulting plot command is 120 characters long.

X = character X (character 34 in the CHAR94 table)

Y = character Y (character 35 in the CHAR94 table)

s_x = + or - character depending upon the sign of the x-coordinate.

s_y = + or - character depending upon the sign of the y-coordinate.

x_i = x-coordinate digit.

y_i = y-coordinate digit.

b = blank (character 48 in the CHAR94 table).

c_i = control character.

a = annotation character.

3.4.59.5 Design Requirements

Subroutine used: SWRITE.

3.4.60 WPLT3 (Write a Plotter Command for Plotter 3).

3.4.60.1 Entry Point: WPLT3.

3.4.60.2 Purpose

To write a plot command for plotter 3.

3.4.60.3 Calling Sequence

CALL WPLT3 (A,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

A(1) and A(2) = 36 bit plot command set up by AXIS3, LINE3, or TYPE3, as 2 18-bit words (right justified, leading zeros) - input.

$$\text{ØPT} = \left\{\begin{array}{l} 0, \text{ if A = plot command} \\ 1, \text{ if a series of plot commands is to be terminated} \end{array}\right\} \text{ - integer - input.}$$

/PLTDAT/

PLØT - GINØ file name of the plot tape to be written - BCD - input.

MAXCHR - plot tape buffer size (number of characters) - integer - input.

3.4.60.4 Method

Each plotter command is 36 bits long (6 six-bit characters). Six of the 36 bits in A(1) and A(2) are written on the plot tape until all 36 bits have been written. In addition, the number of six-bit characters written in a record is calculated. When WPLT3 is called with ØPT = 1, a check is made to determine if the number of 6 bit characters written in the current record is an integer multiple of the number of characters per word on the computer. If such is not the case, an additional 36 bit command is written as many times as necessary until this condition does exist before terminating the plot tape record. The command used will do nothing to affect the generated plot.

3.4.60.5 Design Requirements

Subroutine used: SWRITE.

3.4.61 GINØIØ (GINØ Input/Output Routine).

3.4.61.1 Entry Point: GINØIØ.

3.4.61.2 Purpose

GINØIØ executes the physical input/output operations for subroutine GINØ.

3.4.61.3 Calling Sequence

CALL GINØIØ($n,ØPCØDE,BUFF)

CØMMØN/GINØX/LGINØX,FILEX,EØR,ØP,ENTRY,LSTNAM,N,NAME,NTAPE,XYZ(2),UNITAB(75),BUFADD(75),
N3UFF3,ERRØR,NØSECT.

n - FØRTRAN statement number defining return in the event of an I/Ø error.

ØPCØDE - { 1, Rewind; 2, Write one block; 3, Read one block; 4, Backspace one block; 5, Forward space one block } input - integer.

BUFF - Address of the block to be read or written.

FILEX - Unit number of file - integer - input.

NBUFF3 - Length of block to be read or written - integer - input.

ERRØR - { 7, Abnormal completion of I/Ø operation; 8, Physical end-of-file encountered; 9, Data transmission error } integer - output.

NØSECT - Number of sectors per block on FASTRAND drum (Univac 1108 only) - integer - input.

3.4.61.4 Method

The machine cell in /SYSTEM/ is tested. For the IBM 7094 or IBM S/360 computers, FØRTRAN REWIND, WRITE, READ and BACKSPACE operations are used. For the Univac 1108, the NTRAN routine is used.

3.4.61.5 Design Requirements

GINØIØ is designed as an integral part of the GINØ collection of routines and is to be called only by GINØ.

Since all input/output operations by GINØ are made by GINØIØ, a change to interface with a new or different operating system can easily be made by modifying GINØIØ.

3.4.62 EJECT (Automatic Page Eject)

3.4.62.1 Entry Point: EJECT

3.4.62.2 Purpose

Automatic line counting for printed output and new page initiation when pages are filled.

3.4.62.3 Calling Sequence

K = EJECT (LINES)

CØMMØN /SYSTEM/ - see SYSTEM table description, section 2.4.1.8.

where:

LINES - Number of lines to be printed.

/SYSTEM/

MAXLIN - Maximum number of lines permitted per page.

LINCNT - Number of lines thus far printed on this page.

3.4.62.4 Method

If the number of lines already printed on this page (LINCNT) added to the number of lines about to be printed (LINES) would be greater than the number of lines permitted per page (MAXLIN), a new page is started (CALL PAGE1), the current line counter is set to the number of lines to be printed (LINCNT = LINES), and the result of this function is set to 1 (EJECT = 1).

If the number of lines about to be printed (LINES) will fit on this page (LINCNT + LINES $\leq$ MAXLIN), the result of this function is set to 0 (EJECT = 0).

3.4.62.5 Design Requirements

If it is desired to force a new page to be started, simply set LINCNT = MAXLIN before calling this function.

3.4.63 PLAMAT (Material Property Utility for Two-Dimensional Elements in Piecewise Linear Analysis).

3.4.63.1 Entry Point: PLAMAT.

3.4.63.2 Purpose

To perform the following matrix operation:

$$[C] = [A]^T [B] [A] ,$$

where [A] is equal to [U] (see the subroutine description for PREMAT and MAT, section 3.4.36.3, for a definition of [U] with INFLAG = 2), and [B] is equal to a previously calculated material properties matrix which is in common block /PLAGP/. The result [C], which is symmetric, is stored in common block /MATØUT/.

3.4.63.3 Calling Sequence

```
CALL PLAMAT
CØMMØN/PLAGP/GP(9),MIDGP,ELID
CØMMØN/MATIN/MATID,INFLAG,ELTEMP,PLAARG,SINTH,CØSTH
CØMMØN/MATØUT/G11,G12,G13,G22,G23,G33,DUMMY(14)
```

where:

/PLAGP/

GP(9) = 3x3 material properties matrix calculated in a PLA element driver - real - input.

MIDGP = the material identification number associated with GP - integer - input.

ELID = the element identification number associated with GP - integer - input.

/MATIN/

MATID = the incoming material identification number - integer - input.

INFLAG, ELTEMP, PLAARG = not used by PLAMAT.

SINTH = Sine of the material property orientation angle - real - input.

CØSTH = Cosine of the material property orientation angle - real - input.

/MATØUT/

Same as /MATØUT/ with INFLAG = 2 as described in section 3.4.36.3 except only the first six cells are used.

3.4.63.4 Method

This routine checks to see if the incoming material identification number (MATID) is equal to the material identification number (MIDGP) which was used to calculate the material properties matrix stored in /PLAGP/. If they are not equal, this routine calls MAT with INFLAG = 2 and returns to the calling program. This will only happen in combination elements (TRIA1, TRIA2, QUAD1, QUAD2) where there is a different material identification number used for the membrane and plate properties. If they are equal, then the matrix operation described above is performed.

3.4.64 WPLT4 (Write a Plotter Command for Plotters 4 Through 7).

3.4.64.1 Entry Point: WPLT4.

3.4.64.2 Purpose

To write plot commands for plotters 4 through 7.

3.4.64.3 Calling Sequence

CALL WPLT4 (A,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

A(1) = command type (0 = control, 2 = line, 4 = position)

A(2-N) = additional data used to generate the plot commands (contents and length, N, vary with command type) - integer, input.

ØPT = 0, if a plot command is to be generated; 1, if the current command buffer is to be terminated - integer, input.

/PLTDAT/

PLTMDL = plotter model number - integer - input.

PLØTER = plotter number - integer - input.

PLØT = GINØ file name of the plot tape to be written - BCD - input.

3.4.64.4 Method

The resulting plot command varies in length, depending both on the command type and the amount of necessary drum movement. If A(1) = 0 (control command), A(2) = number of control characters in the resulting command (one character expressed as a right adjusted integer in each word of the A array, starting in A(3)).

If A(1) = 2 or 4 (line or positioning command), the resulting plot command will begin with the characters necessary to lower or raise the pen, respectively, unless the pen is already down or up, respectively. A(2) and A(3) contain the number of X and Y half steps necessary to draw the line

(with the pen down or up), while A(4) and A(5) contain pointers to two character strings needed to draw the entire line except for the final half step. A(6) and A(7) contain the pointers needed to draw the last half step of the line only. The pointers in A(4 - 7) will cause drum movements as follows:

1 = +Y	9 = +Y/2	17 = +X/2, +Y
2 = +X, +Y	10 = +X/2, +Y/2	18 = -X, +Y/2
3 = +X	11 = +X/2	19 = +X, -Y/2
4 = +X, -Y	12 = +X/2, -Y/2	20 = +X/2, -Y
5 = -Y	13 = -Y/2	21 = -X/2, -Y
6 = -X, -Y	14 = -X/2, -Y/2	22 = -X, -Y/2
7 = -X	15 = -X/2	23 = -X, +Y/2
8 = -X, +Y	16 = -X/2, +Y/2	24 = -X/2, +Y

The number of characters in a string is a function of the internal plotter model number, PLTMDL. If PLTMDL = 1, each string is three characters; if PLTMDL = 2 or 4, each string is two characters; and if PLTMDL = 3 or 5, each string is only one character.

As required, this subroutine will automatically initiate each plot tape record with the necessary "conditioning, synchronizing, and start plot" characters, and terminate each plot tape record with the necessary "stop plot" characters.

3.4.64.5 Design Requirements

The only incremental drum movements available for the CALCOMP drum plotter indicated as PLTMDL = 1 are the first eight (8) as listed above. Therefore, when A(1) = 2 or 4, the values in A(4 - 7) must be less than nine (9).

Subroutine used: SWRITE.

3.4.65 WPLT9 (Write a Plotter Command for Plotter 9).

3.4.65.1 Entry Point: WPLT9.

3.4.65.2 Purpose

To write a plot command for plotter 9.

3.4.65.3 Calling Sequence

CALL WPLT9 (A,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

A = 10-character plot command (1 character per word, right justified, leading zeros) - integer - input.

ØPT = { 0 if A = plot command
      { 1 if a series of plot commands is to be terminated } integer - input.

/PLTDAT/

PLØT = GINØ file name of the plot tape to be written - BCD - input.

3.4.65.4 Method

If ØPT = 0, the 10 characters are written on the plot tape without any changes. If ØPT = 1, two characters are appended to the current record:

62_8 (EXIT code) and 61_8 (NØP code).

3.4.65.5 Design Requirements

Subroutine used: SWRITE.

3.4.66 WPLT10 (Write a Plotter Command for the NASTRAN General Purpose Plotter).

3.4.66.1 Entry Point: WPLT10.

3.4.66.2 Purpose

To write the plotter commands for the NASTRAN general purpose plotter.

3.4.66.3 Calling Sequence

CALL WPLT10 (A,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

A(1) = plot mode index
A(2) = control index
A(3) = x_1 = x-coordinate
A(4) = y_1 = y-coordinate
A(5) = x_2 = x-coordinate
A(6) = y_2 = y-coordinate

} - integer - input.

ØPT = { 0 if A = plot command
1 if a series of plot commands is to be terminated } - integer - input.

/PLTDAT/

EDGE = size of the borders (x,y) in plotter units - real - input.

PLØT = GINØ file name of the plot tape to be written - BCD - input.

MAXCHR = plot tape buffer size (number of characters) - integer - input.

3.4.66.4 Method

Each plot command written is composed of 30 six-bit unsigned integers. The plot mode index, A(1), and the control index, A(2), are the first two integers. The next 20 integers represent the values in A(3-6). Each value is represented by five 6-bit integers, each integer being a decimal digit of the decimal representation of the value as follows:

$$d_4 d_3 d_2 d_1 d_0$$

where the original integer value is given by

$$d_0 \cdot 10^4 + d_1 \cdot 10^3 + d_2 \cdot 10^2 + d_3 \cdot 10^1 + d_4 \cdot 10^0 \ .$$

This representation is used so as to make it easy to recover the original integer values on any binary computer. The last 8 characters are always zeros.

The end result is a plot command of the following format:

$$MCP_4P_3P_2P_1P_0Q_4Q_3Q_2Q_1Q_0R_4R_3R_2R_1R_0S_4S_3S_2S_1S_000000000$$

where:

M = plot mode index

C = control index

P_i = decimal digit of the 1st integer value

Q_i = decimal digit of the 2nd integer value

R_i = decimal digit of the 3rd integer value

S_i = decimal digit of the 4th integer value

0 = zero

When WPLT10 is called with ØPT = 1, the current plot tape record is filled with as many dummy plot commands as is necessary to generate a fixed length record. The dummy plot command is made of 30 zeros. This is done so that the plot tape can be read in FØRTRAN without having to worry about variable length records as long as the plot tape buffer size (MAXCHR) is an integer multiple of the number of characters per word on the computer on which the plot tape is being read (see section 6.10.6 for further details).

3.4.66.5 Design Requirements

Subroutine used: SWRITE.

3.4.67 PLTSET (Plotting Parameter Initialization).

3.4.67.1 Entry Point: PLTSET.

3.4.67.2 Purpose

Given the internal plotter and model numbers, to initialize the /XXPARM/ and /PLTDAT/ tables as needed by the NASTRAN plotter software package.

3.4.67.3 Calling Sequence

```
CALL PLTSET
```

CØMMØN/XXPARM/ - see XXPARM Miscellaneous Table description, section 2.5.

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

/XXPARM/

PBUFSZ = plot tape buffer size (number of words) - integer - output.

PAPSIZ = size of the paper to be used (inches) - real - input.

/PLTDAT/

MØDEL = internal plotter model number - integer - input.

PLØTER = internal plotter number - integer - input.

REG = plotting region parameters - real - output.

AXYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - output.

XYEDGE = size of the borders (x,y) in plotter units - real - output.

XYMAX = maximum useable x and y coordinate values on the plotter - real - output.

CNTSIN = number of plotter counts per inch of paper - real - output.

CNTCHR = number of plotter counts per character in the x and y directions - real - output.

PLTYPE = plotter type - integer - output.

PBFSIZ = plot tape buffer size (number of characters) - integer - output.

3.4.67.4 Method

Using the internal plotter (PLØTER) and model (MØDEL) numbers, the initialization needed to properly use the NASTRAN plotting software is performed as follows:

1. Section 2 of /PLTDAT/, of which XYMAX, CNTSIN, CNTCHR, PLTYPE and PBFSIZ are a part, is set to a duplicate of section PLØTER+2.

2. PBUFSZ of /XXPARM/ is then set to PBFSIZ/CHRWRD where CHRWRD = number of characters per word on the subject computer.

3. AXYMAX and XYEDGE are calculated based upon the plotter type and/or paper size. If the plotter is a table plotter (PLTYPE = +2 or -2), the borders are set up as 1/2 inch borders. If the plotter is not a table plotter and has no typing capability (PLTYPE = -1 or -3), the borders are set up as half the horizontal and vertical character sizes (CNTCHR/2). Otherwise, the borders are set to zero.

4. The plotting region is then set to (0,0,AXYMAX(1),AXYMAX(2)). This region can be subsequently altered by the module writer.

3.4.68 DRWCHR (To Draw a Line of Characters).

3.4.68.1 Entry Point: DRWCHR.

3.4.68.2 Purpose

To draw a line of characters on a plotter, horizontally or vertically.

3.4.68.3 Calling Sequence

CALL DRWCHR (X,Y,XYD,CHR,NN,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

CØMMØN/CHRDRW/ - see CHRDRW Miscellaneous Table description, section 2.5.

where:

X,Y = starting or ending coordinate of the line of characters to be drawn (always left-to-right or top-to-bottom) - real - input.

XYD = $\left\{\begin{array}{l}\pm 1 \text{ if X = starting or ending point of the line} \\ \pm 2 \text{ if Y = starting or ending point of the line}\end{array}\right\}$ - integer - input.

CHR = indices of the line of characters to be drawn (see subroutine TiPE) - integer - input.

NN = number of the characters to be drawn - integer - input.

ØPT = $\left\{\begin{array}{l}-1 \text{ to initiate the line mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to draw a line of characters.}\end{array}\right\}$ - integer - input.

/PLTDAT/

REG = plot region parameters - real - input.

XYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - input.

EDGE = size of the border (x,y) in plotter units - real - input.

CNTCHR = number of plotter counts per character in the x and y directions - real - input.

/CHRDRW/

LSTIND = index of the last character which can be drawn - integer - input.

CHRIND = indices into XYCHR used to locate the data needed to draw characters - integer - input.

XYCHR = lines which must be drawn to produce alphanumeric characters - integer - input.

3.4.68.4 Method

If ØPT = 0, all other arguments are ignored and LINE is called. Otherwise, the characters are drawn. The width and height of each character position are assumed to be integer multiples of 8 and 16, respectively. The size of the drawn character will be this integer multiple of 6. The remaining space in each character position is used as the horizontal and vertical spacing. No character will be drawn outside the region specified in REG.

3.4.68.5 Design Requirements

Subroutine used: LINE.

3.4.69 FNDPLT (Determine the Internal Plotter and Model Indices).

3.4.69.1 Entry Point: FNDPLT.

3.4.69.2 Purpose

Given the external name and model of a plotter, to determine the corresponding internal plotter and model numbers used by the NASTRAN plotting software package.

3.4.69.3 Calling Sequence

CALL FNDPLT (PLØTER,MØDEL,PLTNAM,PMØDEL)

where:

PLØTER = internal plotter number - integer - output.

MØDEL = internal model number - integer - output.

PLTNAM(2) = external plotter name - BCD - input.

PMØDEL(2) = external model name - integer or BCD - input and output.

3.4.69.4 Method

PLTNAM and PMØDEL are compared with an internal table of plotter names and models. When a match is found, PLØTER and MØDEL are set to the corresponding internal plotter and model numbers. If a match is found only for the plotter name (PLTNAM), the model name for the first model appropriate to the matched model name will be used to determine PLØTER and MØDEL, and the model name used will be stored in PMØDEL. If no match is found, PLØTER and MØDEL will be set to zero. See section 3.1 for further details.

3.5 MATRIX SUBROUTINE DESCRIPTIONS.

3.5.1 BLDPK (Build a Packed Column of a Matrix).

3.5.1.1 Entry Points: BLDPK, BLDPKI, ZBLPKI, BLDPKN.

3.5.1.2 Purpose

To write a column of a matrix in NASTRAN packed format.

3.5.1.3 Calling Sequence

If several different matrices are to be packed concurrently, the multi-column version is used:

```
CALL BLDPK(TYPIN,TYPØUT,NAME,BLØCK,ØPT,1)
CALL BLDPKI(A,I,NAME,BLØCK,ØPT)
CALL BLDPKN(NAME,BLØCK,ØPT,MCB)
```

where:

BLDPK is an initialization call and is made once for each column to be packed.

BLDPKI is the call made to supply a single element of the column to be packed.

BLDPKN is a call to terminate processing of the column.

TYPIN - Arithmetic type of the elements to be packed (1 = real single precision, 2 = real double precision, 3 = complex single precision, 4 = complex double precision)- integer - input.

TYPØUT - Arithmetic type of the elements in the packed column - integer - input.

NAME - { ØPT = 'WRITE': GINØ file name of data block where packed column will be written.
{ ØPT = 'WRTCØR': An array in core where packed column will be written.

BLØCK - An array of dimension ≥ 6 for use by BLDPK and BLDPKI.

ØPT - { 'WRITE': The packed column will be written by GINØ. } Input
{ 'WRTCØR': The packed column will be written in core. } Subroutine name

A - An array of dimension 1, 2 or 4 (depending on TYPIN) where the element to be packed is stored - real - input.

I - Row position of element to be packed - integer - input.

MCB - An array of dimension 7 where the trailer information about the matrix is accumulated.

If only one matrix is being packed, the single column version should be used as it is more efficient.

```
CØMMØN/ZBLPKX/A(4),I
```

```
CALL BLDPK(TYPIN,TYPØUT,NAME,0, ØPT,0)
CALL ZBLPKI
CALL BLDPKN(NAME,0,ØPT,MCB)
```

where:

BLDPK and its arguments are as defined above.

ZBLPKI is the call made to provide an element of the column to be packed. The element (A), and its row position (I), are stored in /ZBLPKX/ by the user prior to each CALL ZBLPKI.

BLDPKN and its parameters are defined as above.

Note:

BLDPKN accumulates the following two words of MCB:

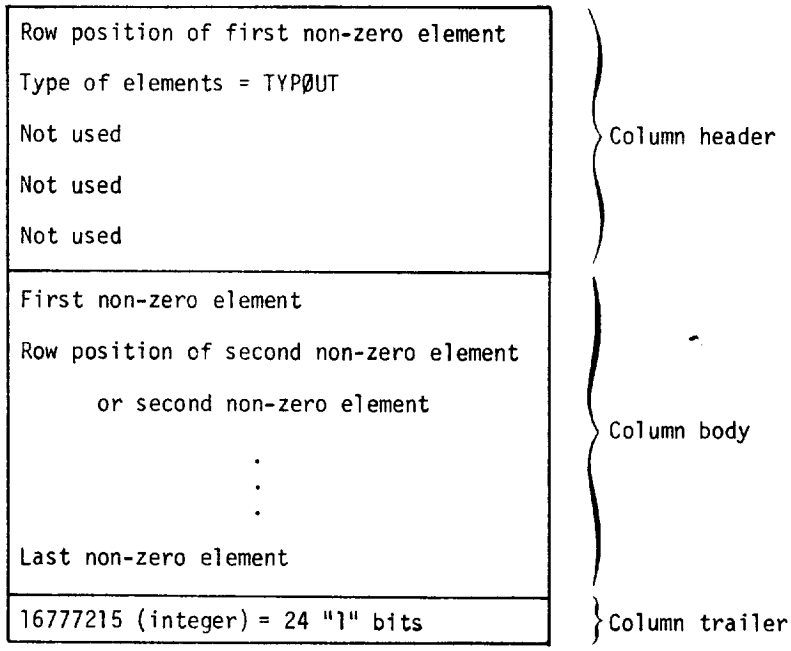
MCB(2) = column number

MCB(6) = number of words in the densest column

In the multi-column version, BLØCK must be different for each matrix being packed.

3.5.1.4 Method

Format of a packed column (one logical record):



Example:

Assume a column of a real single precision 19xN matrix is to be packed in real single precision form:

Row Position	Value	Packed Format
1	0.0	4
2	0.0	1
3	0.0	Not used
4	1.0	Not used
5	2.0	Not used
6	0.0	1.0
7	3.0	2.0
8	4.0	7
9	5.0	3.0
10	6.0	4.0
11	7.0	5.0
12	8.0	6.0
13	0.0	7.0
14	0.0	8.0
15	0.0	16777215
16	0.0	
17	0.0	
18	0.0	
19	0.0	

For each of the 16 possible combinations of TYPIN and TYPØUT, BLDPK sets switches for the type of conversion to be used in packing. For the multi-column version, these switches and other pointers are saved in BLØCK. BLDPKI restores the switches and pointers and moves the element and its row position to /ZBLPKX/ then enters code common with ZBLPKI. If the element is zero, an immediate return is given. If the element is the first non-zero element for the column, the 5-word header is written followed by the element. Otherwise, the row position of the current element is compared to the row position of the last element. If the difference is unequal to one, the current row position is written. In either case, the non-zero element is written and return is made.

3.5.1.5 Design Requirements

Let I_j and I_{j+1} be the rows positions of two elements supplied in successive calls to BLDPKI or ZBLPKI. Then $I_{j+1} > I_j$ for all j of a column.

If ØPT = 'WRTCØR', then the block NAME must be initialized prior to each call to BLDPK (see ØPNCØR).

If the single column version is used, subroutine PACK may not be called between calls to BLDPK and BLDPKN.

MCB(2) and MCB(6) must be set to zero by the user prior to the first call of BLDPKN for a matrix.

3.5.2 PACK (Pack a Column of a Matrix).

3.5.2.1 Entry Point: PACK.

3.5.2.2 Purpose

To pack and write a column of a matrix.

3.5.2.3 Calling Sequence

```
COMMON/PACKX/TYPIN,TYPOUT,I,N,INCR
CALL PACK(A,NAME,OPT,MCB)
```

A - An array where the elements of the column are stored in unpacked form.

NAME - { OPT = 'WRITE', GINO name of the data block where the packed column will be written.
OPT = 'WRTCOR', an array in core where the packed column will be stored.

OPT - { WRITE: Packed column will be written by GINO.
WRTCOR: Packed column will be stored in core at NAME. } Input Subroutine name

MCB - An array of dimension = 7 where the matrix trailer information will be accumulated.

TYPIN - Arithmetic type of the elements of the column stored at A (1 = real single precision, 2 = real double precision, 3 = complex single precision, 4 = complex double precision).

TYPOUT - Arithmetic type in which the elements are to be in packed form. Same convention as TYPIN.

I - Row position of the element stored at A(1).

N - Row position of the last element in the column stored at A.

INCR - Spacing of the elements in column stored at A in units of elements, e.g., if real double precision elements are stored consecutively, INCR = 1.

3.5.2.4 Method

BLDPK is called to initiate activity for the column. For each element in the column, ZBLPKI is called to perform packing and writing. BLDPKN is called to terminate activity for the column and update the matrix trailer.

3.5.2.5 Design Requirements

See subroutine description for BLDPK, section 3.5.1.

3.5.3 INTPK (Interpret a Packed Column of a Matrix).

3.5.3.1 Entry Points: INTPK, INTPKI, ZNTPKI.

3.5.3.2 Purpose

To read and interpret a column of a matrix in NASTRAN packed format.

3.5.3.3 Calling Sequence

If several different matrices are to be read and interpreted concurrently, the multi-column version is used.

CALL INTPK($n,NAME,BLØCK,ØPT,TYPØUT,1)

CALL INTPKI(A,I,NAME,BLØCK,ØPT,EØL)

where INTPK is the initialization call and is made once for each column to be read and interpreted.

INTPKI is the call made to read successive non-zero elements of the column. Each call to INTPKI returns one non-zero element.

n - FØRTRAN statement number defining return to be taken in the event the column is null.

NAME { ØPT = 'READ', GINØ file name of data block where the matrix is stored.
{ ØPT = 'RDCØR', as array of core storage where the matrix is stored.

BLØCK - An array of dimension ≥ 6 for use by INTPK and INTPKI.

ØPT { 'READ' - The column will be read by GINØ.
{ 'RDCØR' - The column will be read from core storage. } Input, Subroutine name

TYPØUT - Arithmetic type into which the elements are to be unpacked (±1 = real single precision, ±2 = real double precision, ±3 = complex single precision, ±4 = complex double precision). If TYPØUT < 0, the sign of each non-zero element is to be changed - integer - input.

A - An array of dimension 1, 2 or 4, depending on TYPØUT, where the non-zero element is to be stored - real - output.

I - Row position of the non-zero element - integer - output.

EØL = 1 indicates last non-zero element in the column was read on the current call to INTPKI, =0 otherwise - integer - output.

If only one matrix is to be read and interpreted, the single-column version should be used as it is more efficient.

```
CØMMØN/ZNTPKX/A(4),I,EØL,EØR
CALL INTPK($n,NAME,0,ØPT,TYPØUT,0)
CALL ZNTPKI
```

where INTPK and its arguments are defined as above.

ZNTPKI is the call made to read successive non-zero elements of the column. One element (A), its row position (I), end-of-column indicator (EØL), and end-of-record indicator (EØR) are stored in /ZNTPKX/ for each call to ZNTPKI.

EØL is defined as above.

EØR = 1 indicates the end-of-record has been read by ZNTPKI, = 0 otherwise (ZNTPKI buffers ahead so that EØR will usually be one before EØL is one. EØR is always one when EØL = 1).

3.5.3.4 Method

INTPK reads the 5-word line header for the column. If ØPT exits via RETURN 2, a null column exists and RETURN 1 is made to the user. Otherwise, for each of the 32 combinations of TYPØUT and the type of the elements in the column, a switch for pickup and conversion of the elements is set. For the multi-column version, this switch and other pointers are stored in BLØCK. For the single column version, one buffer is read into /DESCRP/. INTPKI restores the switch and pointers and then enters code common with ZNTPKI. A test is made to determine if a read is necessary (this is almost always required in the multi-column version since only one element at a time is read). The non-zero element is picked up from /DESCRP/, converted if necessary, and stored in /ZNTPKX/. Its row position is stored in /ZNTPKX/. The next word in the column is now tested (a read being given first if necessary). If the (integer) absolute value of this number is less than 16777215, the number is the row position of the next non-zero element. If = 16777215, it is the trailer word and the last non-zero element has been read. In this case EØL is set to 1. If > 16777215, the number is a real number and the row position of the next non-zero element equals the current row position plus one. For the multi-column version, A, I and EØL are moved from /ZNTPKX/ to the user, and the parameters are saved in BLØCK.

3.5.3.5 Design Requirements

1. If ØPT = RDCØR, the calling module must initialize NAME prior to each call to INTPK see ØPNCØR (section 3.4.13).

2. If the single column version is used, subroutine UNPACK must not be called during interpretation of a column, i.e., subsequent to a call to INTPK and prior to a return from ZNTPKI with EØL = 1.

3. The format of floating point words on computers which execute this program must be such that any non-zero floating point word is larger in absolute value than the integer 16777215.

4. When using the single column version, if the user does not complete interpretation of the column, he must insure that the remainder of the column is skipped. This may be accomplished by

```
IF (EØR.EQ.0) CALL FWDREC($n,NAME).
```

5. The first word of /DESCRP/ must contain the length of /DESCRP/.

3.5.3.6 Diagnostic Messages

The following messages may be issued by INTPK:

3002

3003

3.5.4 UNPACK (Unpack a Packed Column of a Matrix).

3.5.4.1 Entry Point: UNPACK.

3.5.4.2 Purpose

To read and unpack a column of a matrix stored in NASTRAN packed format.

3.5.4.3 Calling Sequence

```
CALL UNPACK($n,NAME,A,ØPT)

COMMØN/UNPAKX/TYPØUT,I,N,INCR
```

n - FØRTRAN statement number defining return to be taken if the column is null.

NAME - { ØPT = 'READ', GINØ name of data block containing the column to be unpacked.
       { ØPT = 'RDCØR', an array where the packed column is stored.

ØPT = { 'READ' - column will be read by GINØ.
      { 'RDCØR' - column will be read from core at NAME. } Subroutine name

A - An array where the unpacked column will be stored.

TYPØUT - Arithmetic type in which the elements are to be stored at A (1 = real single precision, 2 = real double precision, 3 = complex single precision, 4 = complex double precision). TYPØUT ≤ 0 means that each of the elements will be stored with a change of sign.

I - Row position of the element to be stored at A(1).

N - Row position of the last element to be stored at A.

INCR - Spacing of the elements to be stored at A in units of elements, i.e., if complex single precision elements are to be stored at A(1), A(5), A(7), etc., INCR = 2.

Notes:

1. Zeros are stored for zero elements.

2. If I ≤ 0 or N ≤ 0, the column is unpacked from the first non-zero element through the last non-zero element and I and N are set to these row positions.

3. If return to statement n is given, zeros are <u>not</u> stored at A.

3.5.4.4 Method

Activity for the column is initiated by a call to INTPK. A non-standard return from INTPK results in an immediate RETURN 1 to the user. ZNTPKI is called to obtain the first non-zero element. If its row position is less than I, ZNTPKI is repeatedly called until a row nosition ≥ I or end-of-column is found. If the row position of the first non-zero element is greater than I, zeros are stored for the missing elements. Each non-zero element whose row position is less than or equal to N is stored and zeros are stored for missing elements. Non-zero elements whose row positions are greater than N are skipped until the end-of-column is reached.

3.5.4.5 Design Requirements

See subroutine INTPK (see section 3.5.3).

3.5.4.6 Diagnostic Messages

In a coding sense messages 3002 and 3003 are possible. However, they violate the design of GINØ or ØPNCØR and therefcre, if obtained, should indicate an obscure program design error or machine error.

3.5.5 CALCV (Compute a Partitioning Vector).

3.5.5.1 Entry Point: CALCV.

3.5.5.2 Purpose

To build a partitioning vector of zeros, ones and twos to be used by subroutines MERGE and PARTN.

3.5.5.3 Calling Sequence

```
CALL CALCV(FILEP,SET1,SUB0,SUB1,CØRE)
```

FILEP - GINØ file number of partitioning vector - integer - input.

SET1 - Bit position of major set - integer - input.

SUB0 - Bit position of zero subset - integer - input.

SUB1 - Bit position of one subset - integer - input.

CØRE - Open core.

```
CØMMØN/PATX/LCØRE,NSUB0,NSUB1,NSUB2,FUSET
```

LCØRE - Length of open core - integer - input.

NSUB0 - Number of rows in zero subset - integer - output.

NSUB1 - Number of rows in one subset - integer - output.

NSUB2 - Number of rows in two subset (not in one or zero subset) - integer - output.

FUSET - File name of USET - integer - input.

3.5.5.4 Method

Each element of USET is examined and classified. If it belongs to SET1 it is further classified into SUB0, SUB1, and SUB2.

A vector is constructed which has zeros, ones and twos in order as elements of USET are so classified.

3.5.5.5 Design Requirements

LCØRE must be $\geq$ twice length of GINØ buffer.

3.5.5.6 Diagnostic Messages

System messages if USET or FILEP are not correct GINØ files.

3.5.6 PARTN - MERGE (Partition a Matrix - Merge Matricies Together).

3.5.6.1 Entry Point: PARTN, MERGE. PARTN and MERGE are two distinct routines but are so closely related that they are described together here.

3.5.6.2 Purpose

PARTN will break up a matrix into four submatrices.

$$[A] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

MERGE is the inverse of PARTN in that given the four building blocks $A_{11},\ldots,A_{22}$ MERGE will reconstruct [A].

3.5.6.3 Calling Sequence

CALL PARTN (RP,CP,Z)

CALL MERGE (RP,CP,Z)

RP - Matrix control block of the row partitioning vector - integer - input.

CP - Matrix control block of the column partitioning vector - integer - input.

Z - Array of open core.

If RP(1) $\leq$ 0, or CP(1) $\leq$ 0, the core locations from RP(8) or CP(8) will contain this vector in packed form.

If RP(1) = CP(1) $<$ 0 the arrays RP and CP coincide in core.

COMMON/PARMEG/MCBA(7),MCBA11(7),MCBA21(7),MCBA12(7),MCBA22(7),LCORE,RULE

MCBA - Matrix control block for [A] - input.

MCBA11 - Matrix control block for [A11] - input.

MCBA21 - Matrix control block for [A21] - input.

MCBA12 - Matrix control block for [A12] - input.

MCBA22 - Matrix control block for [A22] - input.

If any submatrix is not desired or does not exist set MCBAij(1) = 0.

LCORE - Length of Z array - integer - input.

RULE - Rule to be applied to the row and column partitioning vectors - integer - input.

3.5.6.4 Method

Each element of [A] is assigned to the appropriate submatrix by the following schemes.

RULE ≥ 0 N = |RULE|

$a_{ij} \in$ [A11] if RP(I) = CP(J) = N

$a_{ij} \in$ [A21] if RP(I) = N, CP(J) $\neq$ N

$a_{ij} \in$ [A12] if RP(I) $\neq$ N, CP(J) = N

$a_{ij} \in$ [A22] if RP(I) $\neq$ N, CP(J) $\neq$ N

RULE < 0 N = |RULE|

$a_{ij} \in$ [A11] if RP(I) $\geq$ N, CP(J) $\geq$ N

$a_{ij} \in$ [A21] if RP(I) $\geq$ N, CP(J) $<$ N

$a_{ij} \in$ [A12] if RP(I) $<$ N, CP(J) $\geq$ N

$a_{ij} \in$ [A22] if RP(I) $<$ N, CP(J) $<$ N

Subroutine RULER (RULE,IP,ZCØNT,ØNCNT,LIST,NRØWP,BUFF,IØPT) is called twice to accomnlish this assignment where

RULE - Rule tc be applied

IP - Either RP or CP

ZCØNT - Number of elements (row or column) assigned to the one class.

ØNCNT - Number cf elements (row or column) assigned tc the two class.

For example, if RULER is analyzing RP and RP(I) = N this element of RP is said to belong to the 1 class in that it will go either to [A11] or [A21].

LIST - A list of zeros and ones. Zero, if the element belongs to the one class. One, if the element belongs to the two class.

NRØWP - Number of rows in IP

BUFF - One GINØ buffer space

IØPT - If IØPT = 1, LIST will be stored 1 number per word. If IØPT = 0, LIST will be packed 32 bits/word.

Non-zero elements are read, classified and output.

3.5.6.5 Design Requirements

Open core must contain n GINØ buffers + 1 column (single precision) of [A] and 1 row/32 of [A], where n = the number of submatrices present plus one.

3.5.6.6 Diagnostic Messages

If insufficient core is available as described above, fatal message 3008 is given.

3.5.7 SSG2A (Driver for PARTN).

3.5.7.1 Entry Point: SSG2A.

3.5.7.2 Purpose

To partition a vector into two subsets (i.e., to be a driver for PARTN).

3.5.7.3 Calling Sequence

CALL SSG2A(VECTØR,PART1,PART2,PVECT)

VECTØR - GINØ file number of vector to be partitioned - integer - input.

PART1 - GINØ file number of major partition - integer - input.

PART2 - GINØ file number of minor partition - integer - input.

PVECT - GINØ file number of partitioning vector - integer - input.

CØMMØN/PATX/XXX,NRØW1,NRØW2

NRØW1 - Number of rows in PART1 - integer - input.

NRØW2 - Number of rows in PART2 - integer - input.

3.5.7.4 Method

The PARTN common block is filled.

Based on the trailer of VECTØR and NRØW1, NRØW2:

$$\left\{\text{VECTOR}\right\} \Rightarrow \left\{\begin{matrix}\text{PART1}\\ \text{PART2}\end{matrix}\right\}$$

3.5.7.5 Design Requirements

Open core is needed at /SSGA2/.

3.5.8 SDR1B (Driver for MERGE).

3.5.8.1 Entry Point: SDR1B

3.5.8.2 Purpose

To drive MERGE forming VECTØR

$$\left\{\text{VECTØR}\right\} = \left\{\frac{\text{PART1}}{\text{PART2}}\right\}$$

3.5.8.3 Calling Sequence

CALL SDR1B(PVECT,PART1,PART2,VECTØR,MAJØR,SUBO,SUB1,USET,IØPT,IYS)

PVECT - GINØ name of partition vector - integer - input.

PART1 - GINØ name of vector which corresponds to SUBO set - integer - input.

PART2 - GINØ name of vector which corresponds to SUB1 set - integer - input

VECTØR - GINØ name of merged vector - integer - input.

MAJØR - Bit position of set of VECTØR - integer - input.

SUBO - Bit position of set of PART1 - integer - input.

SUB1 - Bit position of set of PART2 - integer - input.

USET - GINØ name of USET - integer - input.

IØPT - '0' }
IYS - '0' } These are used in a module specific call to handle the YS data block in a special manner.

3.5.8.4 Method

CALCV is called to obtain partitioning vector.

PARMEG common block is filled.

MERGE is called.

3.5.8.5 Design Requirements

Open core at /SDRB1/.

3.5.9 UPART (Symmetric Partition Driver).

3.5.9.1 Entry Points: UPART, MPART

3.5.9.2 Purpose

To compute a partitioning vector and then perform a series of symmetric partitions. A symmetric partition is such that the row partitioning vector equals the column partitioning vector.

For example:

$$[K_{nn}] = \left[\begin{array}{c|c} K_{ff} & K_{fs} \\ \hline K_{sf} & K_{ss} \end{array}\right]$$

3.5.9.3 Calling Sequence

CALL UPART(USET,SCR1,MAJØR,SUB0,SUB1)

USET - GINØ file number of USET - integer - input.

SCR1 - Scratch file on which the partitioning vector will be written - integer - input.

MAJØR - Bit position within a USET word of the super set (e.g., n set in the above example) - integer - input.

SUB0 - Bit position of the first subset (e.g., f set in the above example) - integer - input.

SUB1 - Bit position of the second subset (e.g., s set in the above example) - integer - input.

CALL MPART(KNN,KFF,KSF,KFS,KSS)

KNN - GINØ name of the matrix to partitioned - integer - input.

KFF,KSF,KFS,KSS - GINØ names of the partition outputs. A zero will cause the respective matrix not to be written.

3.5.9.4 Method

A call to UPART causes CALCV to compute a partitioning vector.

MPART drives PARTN and is called repeatedly to partition several matrices (i.e. KNN, MNN, BNN, K4NN) in a similar symmetric manner using the same partitioning vector.

3.5.9.5 Design Requirements

Open core at /UPARTX/.

3.5.10 ADD (Matrix Addition Routine).

3.5.10.1 Entry Point: ADD.

3.5.10.2 Purpose

To compute [C] = α[A] + β[B] or on option [C] = α[A].

3.5.10.3 Calling Sequence

```
CALL ADD (Z)
```

Z -- Array of core

```
COMMON/ADDX/MCBA(7),MCBB(7),MCBC(7),TYPA,ALPHA(4),LCORE,TYPB,BETA(4)
```

MCBA - Matrix Control Block for [A] - input.

MCBB - Matrix Control Block for [B] - input.

MCBC - Matrix Control Block for [C] - input.

TYPA - Type of Alpha - integer - input.

1 - real single precision

2 - real double precision

3 - complex single precision

4 - complex double precision

ALPHA - α - input - type depends on TYPA.

LCORE - Length of Z array.

TYPB - Type of BETA - integer - input.

BETA - β - input - type depends on TYPB.

3.5.10.4 Method

The type of arithmetic is determined to be the maximum type of [A], [B], [C], α, β.

The non-zero terms of [A] and [B] are processed into [C].

If MCBB(1) = 0, α[A] will be computed.

3.5.10.5 Design Requirements

Z must be at least three GINO buffers long.

If α[A] only is to be computed, MCBB(1) = 0.

3.5.11 SSG2C (Driver for ADD).

3.5.11.1 Entry Point: SSG2C.

3.5.11.2 Purpose

To drive ADD to compute [C] = α[A] + β[B].

3.5.11.3 Calling Sequence

CALL SSG2C (FILEA,FILEB,FILEC,IØP,BLØCK)

FILEA - GINØ file number of [A] - integer - input.

FILEB - GINØ file number of [B] - integer - input.

FILEC - GINØ file number of [C] - integer - input.

IØP - Option flag - integer - input.
IF IØP < 0 the first column of [A] will be added to each column of [B] to give [C].

BLØCK - 11-word array containing coefficients - input.

Word	Type	Meaning
1	Integer	Type of α
2	Real	α
3		α
4		α
5		α
6		Not used
7	Integer	Type of β
8	Real	β
9		β
10		β
11		β

3.5.11.4 Method

The trailers of FILEA and FILEB and BLØCK are used to fill the ADDX common block.

The type of FILEC is the minimum type compatible with α[A] and β[B].

3.5.11.5 Design Requirements

Open core at /SSGC2/.

3.5.12 MPYAD (Matrix Multiplication Routine).

3.5.12.1 Entry Point: MPYAD.

3.5.12.2 Purpose

To evaluate the matrix equation

$$D = \pm [A] [B] \pm [C] \quad \text{or} \quad D = \pm [A]^T [B] \pm [C].$$

3.5.12.3 Calling Sequence

```
CALL MPYAD(Z,Z,Z)

CØMMØN/MPYADX/A(7),B(7),C(7),D(7),NZ,T,SIGNAB,SIGNC,PREC,SCR
```

Z - An area of working storage.

NZ - The number of computer words at Z.

A,B,C - Matrix control blocks for the matrices A, B, C.

If C(1) = 0, C is not used, i.e. $[D] = \pm [A] [B]$ or $\pm [A^T] [B]$.

D - Matrix control block for the product matrix.

D(1) must contain the GINØ file name prior to entry.

D(5) must contain the arithmetic type of the elements of D. MPYAD will accumulate D(2), D(6) and set D(7) = 0.

T $\begin{cases} = 0, \pm [A] [B] \pm [C] \text{ is computed.} \\ \neq 0, \pm [A]^T [B] \pm [C] \text{ is computed.} \end{cases}$

SIGNAB $= \begin{cases} +1, \text{ compute } +[A] [B] \text{ or } +[A]^T [B] \\ -1, \text{ compute } -[A] [B] \text{ or } -[A]^T [B] \end{cases}$

SIGNC $= \begin{cases} +1, \text{ use } + C \\ -1, \text{ use } - C \end{cases}$

Note: If C(1) = 0, SIGNC is ignored.

PREC $= \begin{cases} 1, \text{ perform arithmetic in single precision} \\ 2, \text{ perform arithmetic in double precision} \end{cases}$

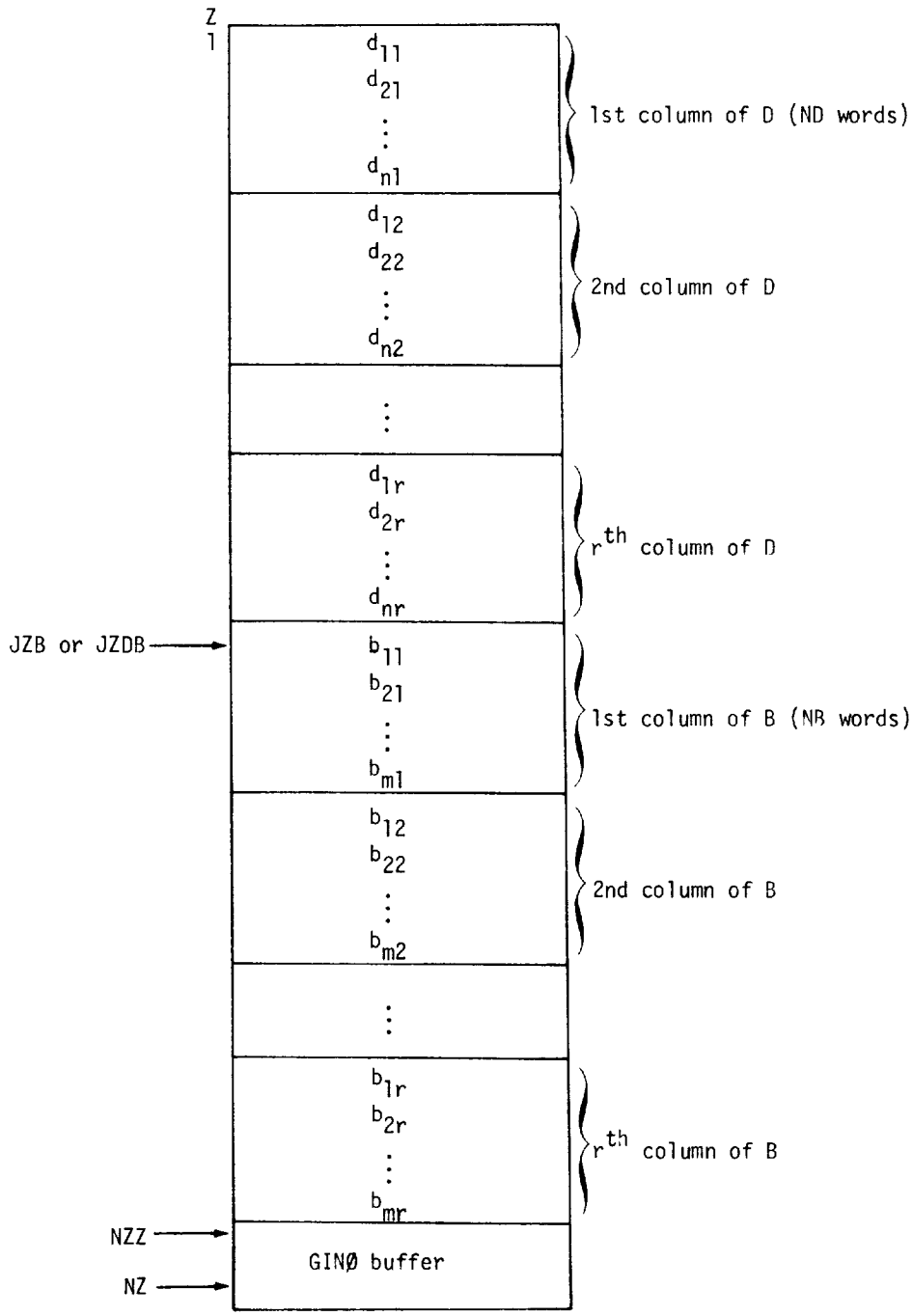
SCR = GINØ file name of a scratch file for use by MPYAD.

3.5.12.4 Method

1. General Comments. Two alternative methods of performing the matrix multiplication are available in MPYAD. Method One holds as many unpacked columns of the B and D matrices as core storage will allow. The A matrix is read interpretively by INTPK. For each non-zero element in A, all combinatorial terms for columns of B currently in core are computed and accumulated in the storage for D. At the completion of one pass of the A matrix, the matrix product is complete to the extent of the number of columns of B currently in core. (If the C matrix is present, columns are initially unpacked into the storage for D.) The process is repeated until the B matrix is exhausted. One GINØ buffer only is required for Method One. The number of passes of the A matrix for Method One equals the number of columns of B divided by the number of columns of B and D which can be held in core at one time. In Method Two one column of B is held in core at one time in packed form (i.e., non-zero terms and row positions only) and either one column of D in unpacked form (T = 0) or one element of D (T = 1). The remaining storage is allocated to storage of columns of A in packed form (as described above). For all the columns of A in storage at one time the B and E matrices are passed, column by column, forming partial answers on each pass. The E matrix is initially the C matrix (if present) and thereafter is the partial product matrix from the previous pass. Three GINØ buffers are required for Method Two. It may be seen that the A matrix is passed once and the number of passes of the B and E matrices equals the number of columns of A divided by number of columns of A that may be held in core at one time.

2. Initialization Phase. The arithmetic type of the elements of D is determined as a function of the types of A, B and C and the precision requested by the user. Various pointers for both Methods One and Two are computed. A determination of sufficient core storage is made (one unpacked column of B plus one unpacked column of D plus one GINØ buffer). Switches are set for the type of arithmetic to be performed and the pickup and conversion of elements of B. If the number of passes for Method One is less than 4, Method One is selected. Otherwise the A matrix is read interpretively once by INTPK, and allocation of storage for Method Two is determined. If the number of passes for Method Two is less than the number for Method One, Method Two is selected. Otherwise Method One is used.

3. Method One. The allocation of core storage for Method One is shown below:



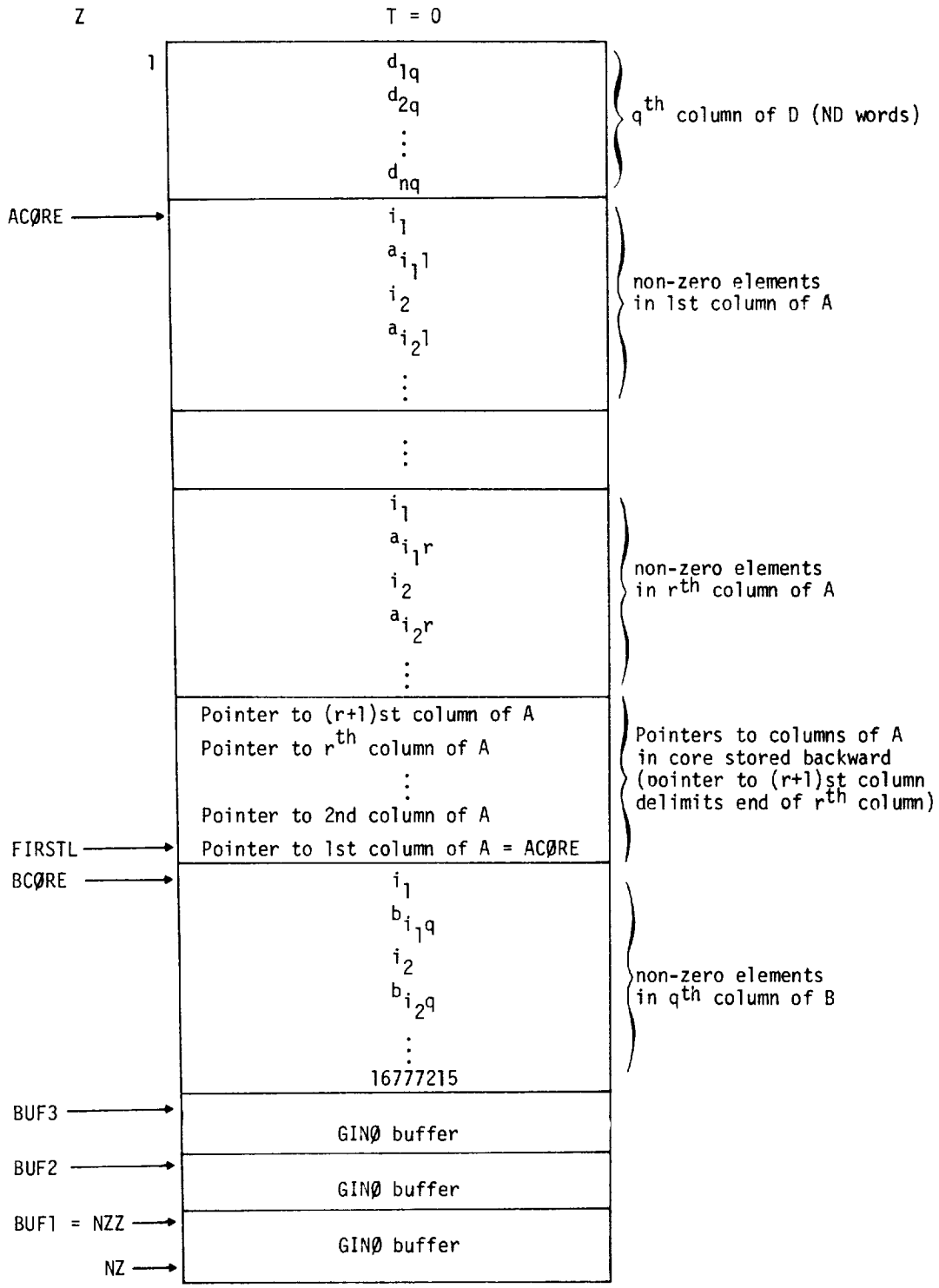
Columns of B and C are read and unpacked by UNPACK. INTPK is called to initiate reading and interpreting the ℓth column of A (ℓ = 1 initially). For each non-zero term in A, $a_{i\ell}$ or $a_{\ell i}$ depending on T, the following arithmetic computations are made:

$$T = 0: \quad d_{ij} = a_{i\ell}\, b_{\ell j} + d_{ij}$$

$$T \neq 0: \quad d_{\ell j} = a_{\ell i}\, b_{ij} + d_{\ell j}$$

where j runs across the columns of B and D currently in core. At the conclusion of a pass of the A matrix, the columns of D in core are packed and written by PACK. The process is repeated until the multiplication is complete.

4. Method Two. The allocation of core storage for Method Two is shown on the next page.



Note: Allocation for case T = 1 is same except that ACØRE = 1. Consequently, additional columns of A may be held in storage on each pass.

To begin each pass, as many columns of A that can be held in core are stored in packed form using INTPK. Pointers to the beginning of each column are kept. For each column of B the following sequence occurs:

1. BLDPK is called to initiate packing of a column of D.
2. The column of B is stored in packed form at BCØRE using INTPK (if column is null, the corresponding C column is copied to D). The transpose flag is tested.
3. $T \neq 0$: for each non-zero term in A in the p^{th} row $d_{pk} = a_{pq}\, b_{qk} + d_{pk}$ is formed. When the row of A is complete, ZBLPKI is called. When all rows of A are complete, BLDPKN is called.
4. $T = 0$: the corresponding column of C is read and unpacked by UNPACK. Then, for each non-zero term in A in the p^{th} column, $d_{kq} = a_{kp}\, b_{pq} + d_{kq}$, is computed. When all columns of A in core are complete, the column of D is packed and written by PACK.
5. When all columns of B and C are complete, a test for completion of the multiplication is made.
6. If incomplete, the C and D files are switched and the process described above is repeated.

3.5.12.5 Design Requirements

Core storage must be sufficient to hold one unpacked column of B plus one unpacked column of D plus one GINØ buffer.

The matrices to be multiplied (and added) must have compatible dimensions. MPYAD does not check this.

3.5.12.6 Information Messages

CØNMSG is called at entry and at exit from MPYAD. Consequently, the line xxxxx MPYAD will appear twice for each call to MPYAD (where xxxxx = time in seconds). The difference is the execution time for MPYAD.

2073 MPYAD METHØD = $\left\{\begin{matrix}1\\2\end{matrix}\right\}$, NØ. ØF PASSES = _____.

This message gives the method selected and number of passes required.

2102 LEFT-HAND MATRIX RØW PØSITIØN _____ ØUT ØF RANGE - IGNØRED.

A term in the A matrix whose row position is larger than the stated dimension was detected and ignored.

3.5.12.7 Diagnostic Messages

The following messages may be issued by MPYAD:

3001

3002

3008

3.5.13 SSG2B (Driver for MPYAD).

3.5.13.1 Entry Point: SSG2B.

3.5.13.2 Purpose

To drive MPYAD to compute

$$[D] = \pm [A] [B] \pm [C]$$

or

$$[D] = \pm [A]^T [B] \pm [C]$$

3.5.13.3 Calling Sequence

CALL SSG2B(FILEA,FILEB,FILEC,FILED,T,PREC,ISIGN,SCR1)

FILEA - GINØ name of [A] - integer - input.

FILEB - GINØ name of [B] - integer - input.

FILEC - GINØ name of [C] - integer - input.

FILED - GINØ name of [D] - integer - input.

T - Transpose flag - integer - input.

T = 0 implies use [A]

T = 1 implies use $[A]^T$

PREC - Precision of computation - integer - input. 1 = real single precision, 2 = real double precision, 3 = complex single precision, 4 = complex double precision.

ISIGN - Sign of products - integer - input.

$$\text{ISIGN} = \pm 1 \Rightarrow \begin{cases} \text{sign [AB]} = \text{sign (ISIGN)} \\ \text{sign [C]} = \text{sign (ISIGN)} \end{cases}$$

$$|\text{ISIGN}| > 1 \Rightarrow + [AB] - [C]$$
$$|\text{ISIGN}| < 1 \Rightarrow - [AB] + [C]$$

SCR1 - GINØ scratch file - integer - input.

3.5.13.4 Method

SSG2B fills /MPYADX/ and calls MPYAD to compute [D] in above equation.

3.5.13.5 Design Requirements

Open core at /SSGB2/.

3.5.14 SDCØMP (Symmetric Decomposition).

3.5.14.1 Entry Point: SDCØMP.

3.5.14.2 Purpose

To decompose a real symmetric matrix [A] into the form [A] = [L] [D] $[L]^T$ where [L] is a unit lower triangular matrix and [D] a diagonal matrix stored in place of the unit elements on the diagonal of [L]. On option, the Cholesky decomposition [A] = [C] $[C]^T$ is done for a real, positive definite matrix, with only the lower triangle [C] being output. SDCØMP will also compute the determinant of [A].

3.5.14.3 Calling Sequence

CALL SDCØMP($$n_1$$,Z,Z,Z)

CØMMØN/SFACT/A(7),L(7),C(7),SCR1,SCR2,NZ,DET,PØWER,CHLSKY

A(7) - Matrix control block for [A].

L(7) - Matrix control block for [L].

C(7) - Matrix control block for $[L]^T$ or [C].

SCR1, SCR2 - Two scratch files.

NZ - The number of computer words at Z.

DET - Double precision cell where the scaled value of the determinant of [A] will be stored.

PØWER - Scale factor to be applied to DET (Determinant = DET*10**PØWER).

CHLSKY - When CHLSKY = 1, form [C]

Z - An area of working storage.

n_1 - Statement number to which control is transferred if the decomposition fails.

3.5.14.4 Method

1. Mathematical Considerations. Any non-singular real symmetric matrix [A] can be uniquely decomposed into the factors [A] = [L] [D] $[L]^T$. The elements of [D] and [L] are given by:

$$d_i = a_{ii} - \sum_{k=1}^{i-1} \ell_{ik} d_k$$

$$\ell_{ij} = [a_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\, d_k\, \ell_{jk}]/d_j$$

2. General Comments. Interpretation of the above equations identifies that element ℓ_{ij} is composed of products of the elements in the i^{th} row of [L] times the elements in the j^{th} row of [L] times the diagonal elements of [D] (Figure 1). Also, element ℓ_{ij} affects only the elements in the i^{th} row or the i^{th} column of [L], (Figure 2).

The above considerations indicate the means of optimizing the decomposition process to perform only the necessary operations and to keep elements in core only as long as needed. As an example, if [A] was strictly banded, with a bandwidth of B, then the inner product calculation of ℓ_{ij} extends only to the band as all other terms are zero (Figure 3). Also in this example, all terms of [L] in column 5 affect only the elements inside the triangle. As soon as these terms are computed, column 5 need not be held in core. The optimum algorithm for banded matrices would be to hold the triangular portion of the band in core, compute terms corresponding to the first column, output the first column, and move the triangular area down the band (Figure 4).

In practice, however, structural matrices are not strictly banded, but semi-banded with a few scattered terms existing outside the band. The basic algorithm remains the same except that the terms outside the band (active elements) must be handled in an analogous manner to those inside the band (band elements). This creates a rectangular storage area for the active rows (rows of [L] which contain at least one active element) plus a second traingular area for terms arising out of interactions between active rows. The storage requirements are shown in Figure 5.

Given a matrix [A], SDCØMP will determine the optimum combination of B (bandwidth), and C (number of active rows) by estimating the time required to decompose [A] for all combinations of B and C and choosing the pair corresponding to the minimum time.

Figures 6, 7, and 8 show how storage is allocated to various operations.

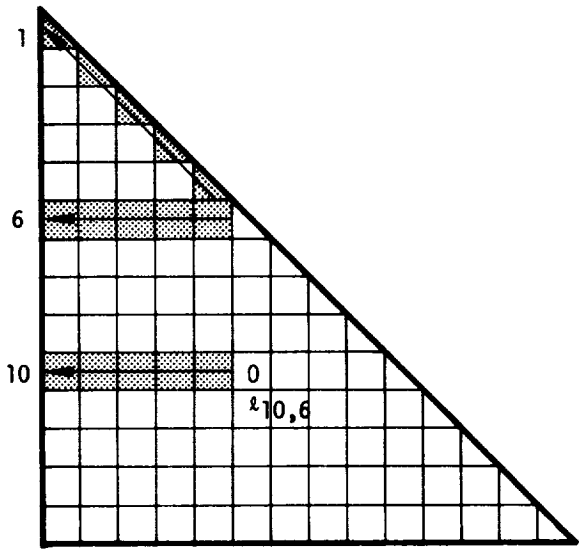


Figure 1. Computation of element ℓ_{ij}:

$$\ell_{10,6} = \left[a_{10,6} - \ell_{10,5}\, d_5\, \ell_{6,5} - \ell_{10,4} d_4 \ell_{6,4} \cdots - \ell_{10,1} d_1 \ell_{6,1}\right] / d_6$$

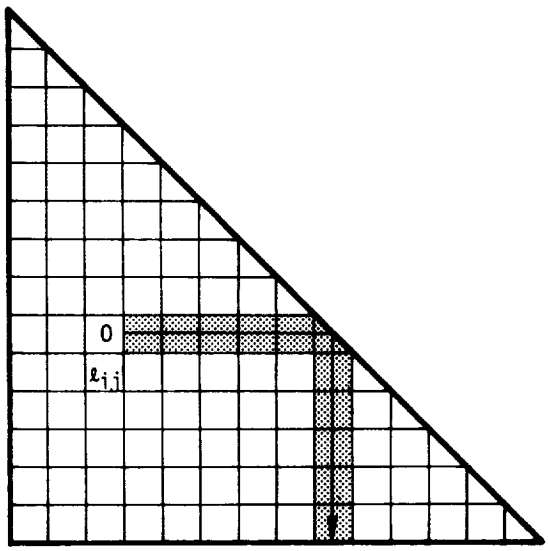


Figure 2. Affect of element ℓ_{ij}.

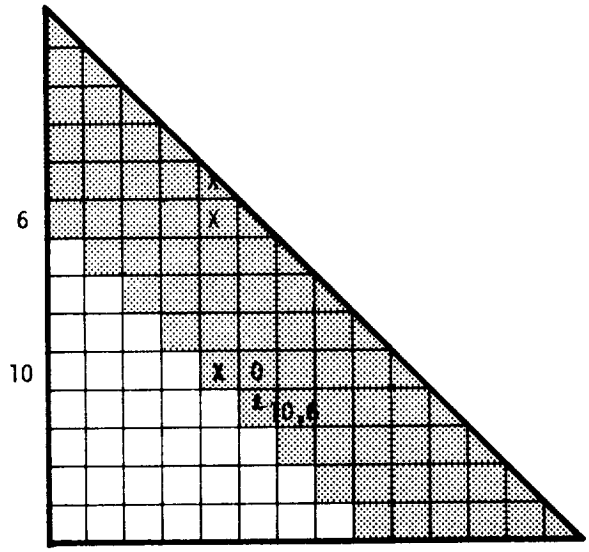


Figure 3. Computation of element ℓ_{ij} (banded matrix):

$$\ell_{10,6} = \left[a_{10,6} - \ell_{10,5}\, d_5 \ell_{6,5}\right] / d_6$$

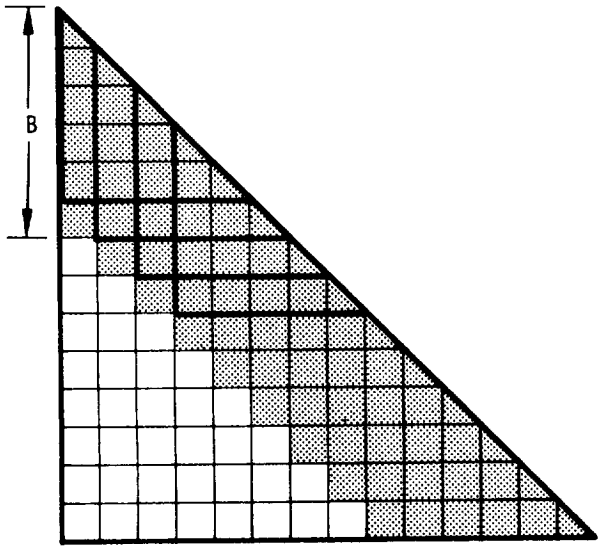


Figure 4. Core storage requirements for a banded matrix.

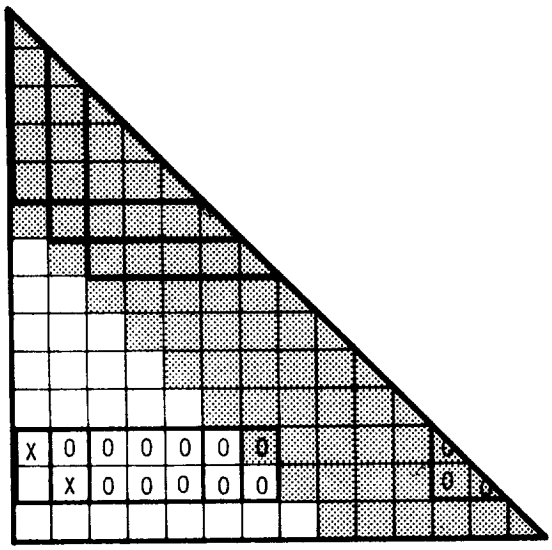


Figure 5. Core storage requirements for a matrix with active rows.

I Storage for the partially computed elements of [L] that are within the band.

II Storage for a completed column of [L].

III Storage for the next column of [A] to be read in.

IV Storage for the partially computed elements of [L] that are outside the band.

V Storage for intermediate results computed from interactions between active rows.

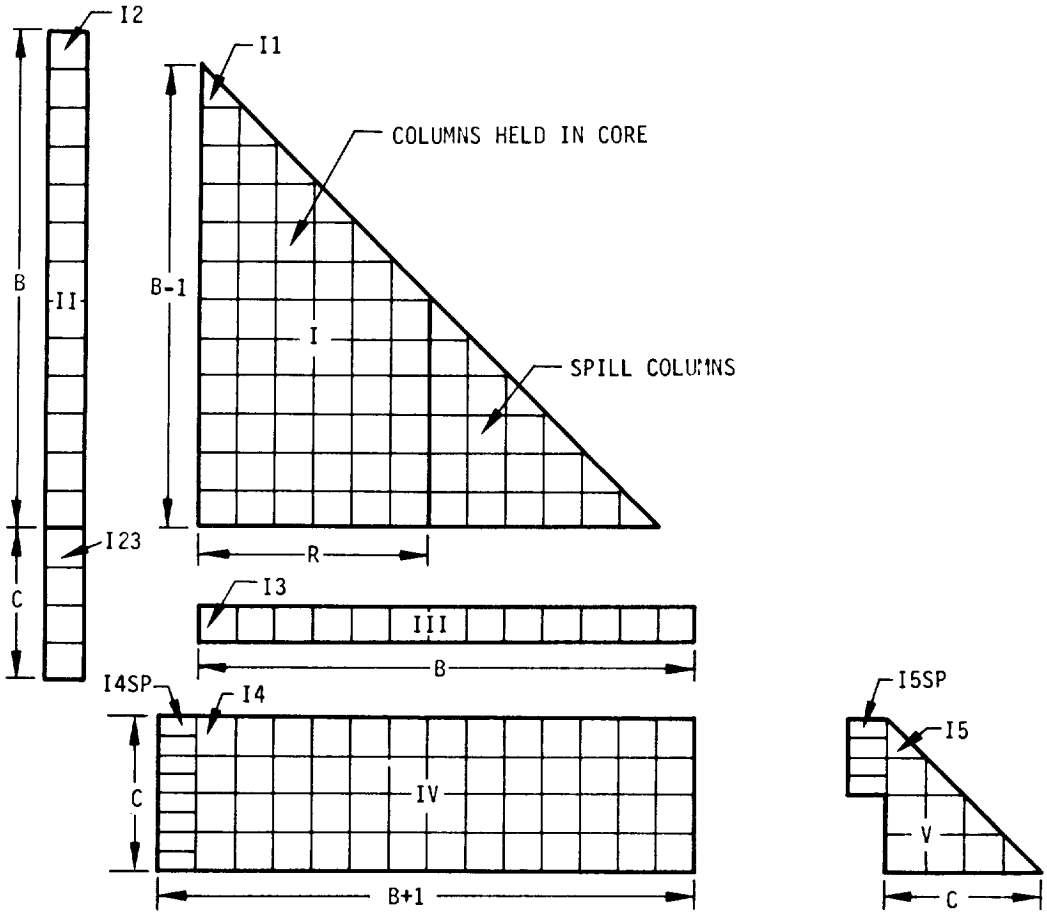


Figure 6. Definition of storage areas.

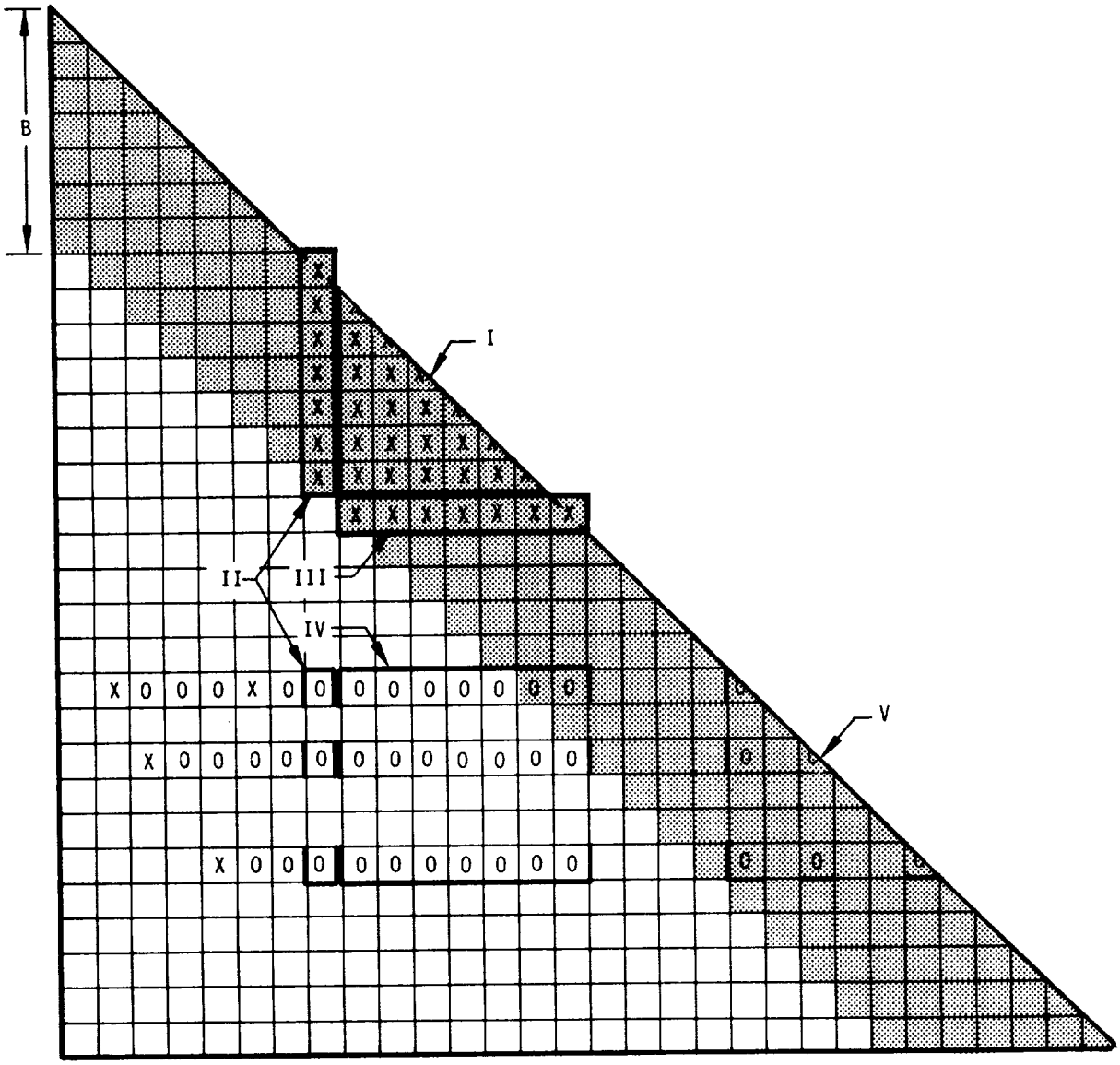


Figure 7. Location of storage areas within the matrix.

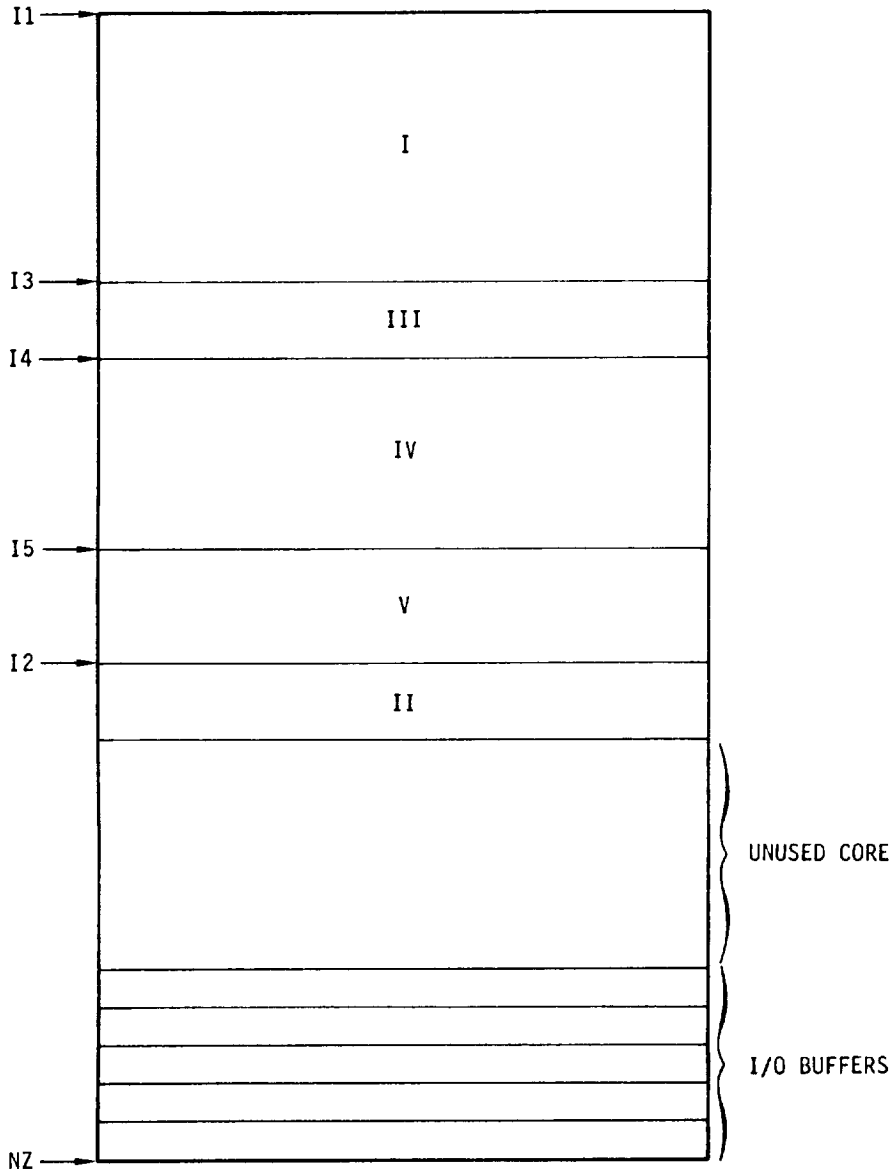


Figure 8. Location of storage areas within core.

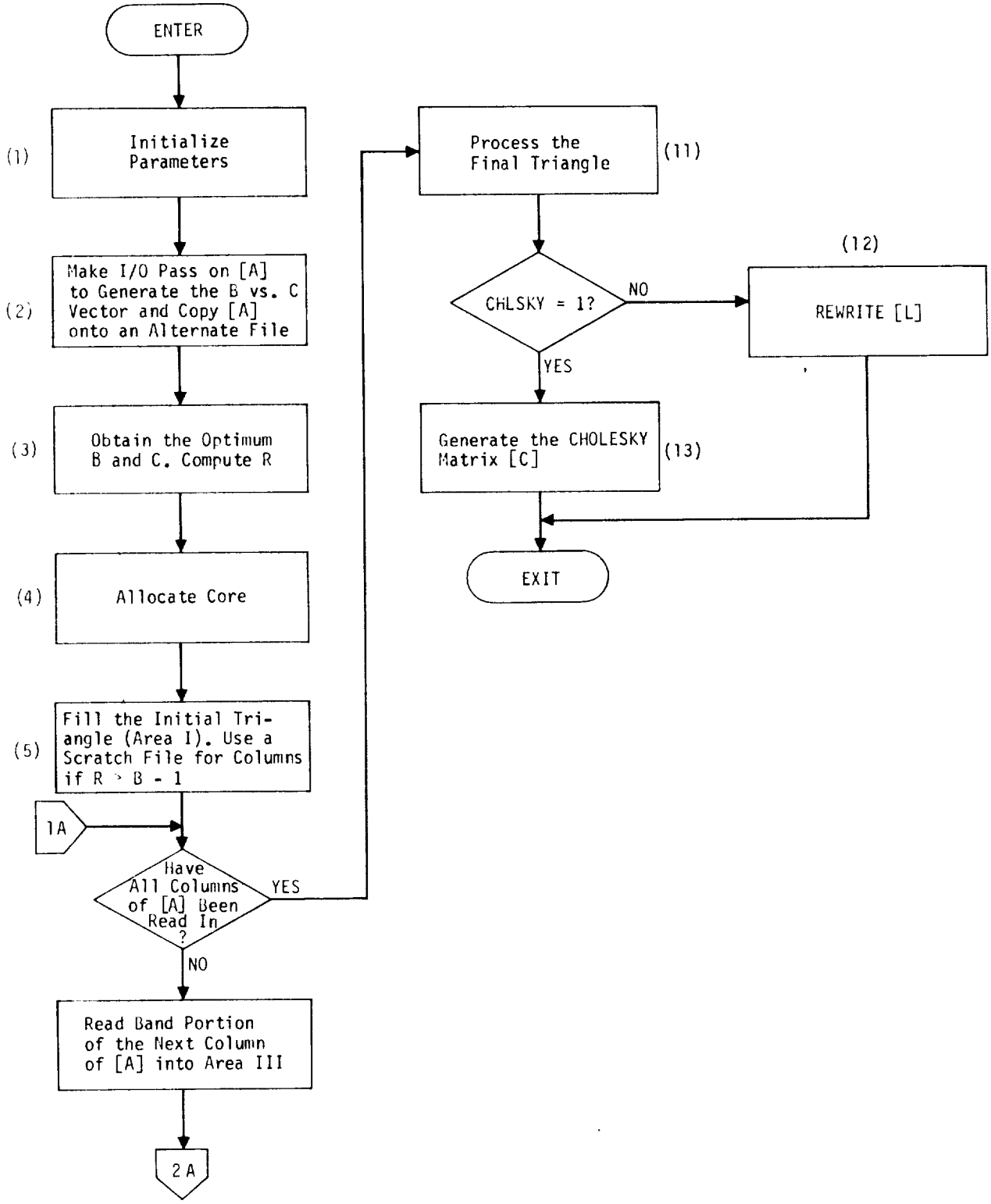


Figure 9.(a) SDCØMP program flow

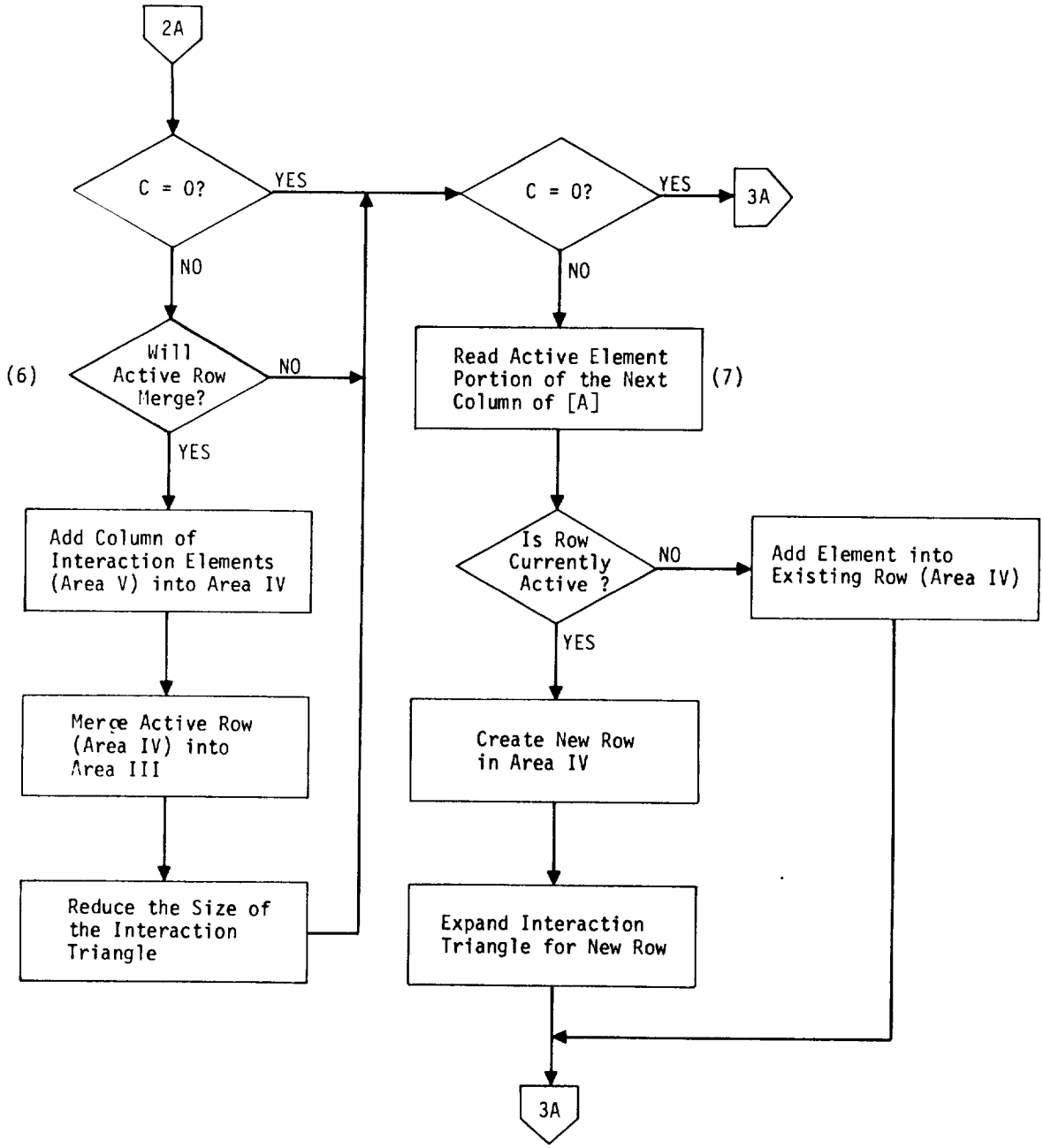


Figure 9.(b) SDCØiiP program flow

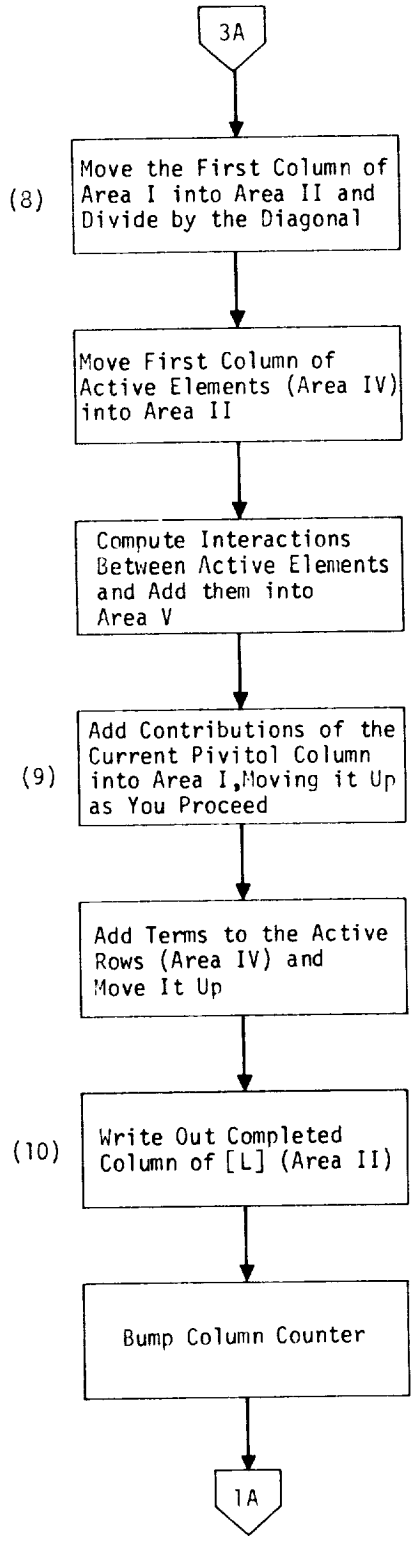


Figure 9.(c) SDCØiP program flow

3. Program Flow. The flowchart in Figure 9 gives the logical flow of SDCØMP. The following are comments expanding on certain areas of the flowchart:

(1) Allocate buffers, initialize determinant, and write header records.

(2) A vector is generated containing the number of active rows for varying bandwidths. The matrix [A] is copied onto an alternate file for use during the decomposition.

(3) The estimated time for decomposition is computed for each combination of B and C. The B and C corresponding to the minimum time is picked. R, the number of columns which can be held in core, is also computed at this time.

(4) Pointers to the various areas of core shown in Figure 6 are computed as a function of B, C and R.

(5) The banded portion of the first B-1 columns of [A] are read into Area I. If $R \leq B-1$, columns are written on a scratch file.

(6) As the decomposition proceeds, active rows gradually merge into the band. This means a column from Area V will merge into Area IV and a row from Area IV will merge into Area III. When this occurs, the current number of active rows decreases by one and the size of the interaction triangle is reduced.

(7) As active elements corresponding to the next column of [A] are read in, they are either added to an already existing active row, or a new active row must be created. Whenever a new row is created, indices are stored identifying the row, the active element is stored in Area IV, and the interaction triangle is expanded to accomodate the added row.

(8) As a column emerges from the triangle in Area I, it is a completed column of [L].

(9) Before the column can be output, all terms involving this column must be computed and stored. These intermediate terms are stored in either Areas I, IV, or V, depending where the elements are located.

(10) A column of [L] can be output. All areas of temporary storage have been updated. Areas I, IV, and III have all moved over one column relative to their previous position in the matrix.

(11) When all columns of [A] have been read in, all active rows have merged and only Area I exists. This final triangle is processed to complete the computation of [L]. If spill exists, as more core is made available by columns of [L] being output, additional columns of the spill file are read in.

(12) The file containing [L] is now complete. An additional file is created with the columns of [L] written in the reverse order. This generates a pseudo upper triangular matrix that is used by FBS for the solution of a set of equations.

(13) If CHLSKY was set, [L] is read, the diagonal term picked up, each column is multiplied by the square root of the diagonal, and the resulting [C] matrix is output.

3.5.14.5 Auxiliary Subroutines

Subroutine Name: LØØP

Purpose: To compute the inner arithmetic loop of SDCØMP.

3.5.14.6 Design Requirements

The input matrix [A] should be well conditioned or positive definite as the decomposition is done without pivoting.

Core storage requirements depend on the parameters B and C. For a given B and C, Areas II, III, IV, and V must reside in core along with a minimum of two columns of Area I and 5 GINØ buffers.

Files containing [L] and $[L]^T$ should be used as input only to FBS as they are not in standard NASTRAN format.

3.5.14.7 Information Messages

1. CØNMSG is called at entry and exit from SDCØMP. The line

xxxx SDCØMP

will appear twice per decomposition. The execution time of SDCØMP will be the difference in the times (where xxxxx = time in seconds).

1. Message 3023 gives the values of the parameters, B, C, and R chosen for the decomposition.
2. Message 3027 gives the estimated time in seconds to do the decomposition.
3. Message 3024 indicates that a matrix has scattered terms way off the diagonal (i.e., a large bandwidth). Instead of searching all combinations of B and C, the search is started at the maximum bandwidth.

3.5.14.8 Diagnostic Messages

1. If SCDØMP was unable to find a combination of B and C which would meet core restrictions, fatal message 3008 occurs.
2. In a coding sense, message 3025 is possible. However, it violates the design of SDCØMP and therefore, if obtained, should indicate an obscure program design error, or machine error.
3. Message 3026 indicates that sufficient space was not reserved for the generation of the B vs. C vector. SDCØMP should be recompiled to increase BMAX and CMAX.

3.5.15 DECØMP (Unsymmetric Matrix Decomposition)

3.5.15.1 Entry Point: DECØMP

3.5.15.2 Purpose

To decompose a real square matrix [A] into the form

$$[A] = [L][U] \tag{1}$$

(where [L] is a unit lower triangular matrix, and [U] is an upper triangular matrix), using partial pivoting within the lower band.

3.5.15.3 Calling Sequence

```
CALL DECØMP ($n,X,X,X)
CØMMØN /DCØMPX/ A(7),L(7),U(7),SCR(3),DET,PØWER,NX,MINDIA,B,BBAR,C,CBAR,R
```

n - Statement number to which control is transferred if [A] is singular.

X - An area of core available to DECØMP.

A - Matrix control block for the input matrix [A] (if A(1) < 0, avoid re-writing [U] in reverse order).

L,U - Matrix control blocks for the output matrices [L] and [U].

SCR(3) - GINØ file names for three scratch files - integer.

DET - Double precision cell where the scaled value of the determinant of [A] will be stored.

PØWER - Scale factor to be applied to DET (det([A]) = DET * 10**PØWER).

NX - Number of computer words available at X.

MINDIA - Double precision word where the value of the minimum diagonal of [U] is stored.

B,BAR, C,CBAR, R - Integer values describing the upper and lower semi-bandwidths, number of activ rows and columns and number of columns of [L] held in core, used to decompose [A]. (If B, BBAR = 0, GENVEC is called to compute the parameters before decomposint [A]. If B, BBAR ≠ 0, the given parameters are used for decompositio

3.5.15.4 Method

1. Mathematical Considerations: By expanding Equation 1, introducing element notation, and forming the multiplication, we can solve for the elements of [L] and [U]. These equations are given by:

$$\ell_{ij} = [a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj}]/u_{jj}, \; i > j \tag{2}$$

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} \ell_{ik} u_{kj}, \; i \leq j \; . \tag{3}$$

2. General Comments: The implementation of the above equations is accomplished with several constraints in mind. The decomposition procedure is optimized such that the minimum number of operations is performed, with the minimum amount of core used. To accomplish this, the elements of the input matrix [A] are separated into two groups: terms inside the band (band elements), and terms outside the band (active elements). Also, pivoting is used only within the lower band to avoid unnecessarily filling the matrix with non-zero terms.

Since, in practice, structural matrices tend to be semi-banded with scattered terms existing outside the band, this division of the matrix should optimize the decomposition process. Several parameters are generated to describe this division. B is defined as the upper bandwidth, $\bar{B}$ as the lower bandwidth, C is defined as the number of active columns, and $\bar{C}$ as the number of active rows, where an active column is defined as a column containing one or more active elements above the diagonal, and an active row contains one or more active elements below the diagonal. Corresponding to these parameters, several storage areas are defined to hold the various parts of the matrix. The description and location of these areas are given in Figures 1, 2 and 3. A flow chart for DECØMP is given in Figure 4.

The storage areas in Figures 1, 2, and 3 are defined as follows:

I Storage for the completed columns of [L] still required for computation.
II Storage for the current column being computed.
III Storage for active column elements.
IV Storage for active row elements.
V Storage for elements created by interactions between active row and column elements.
VI Storage for indexes identifying active columns.
VII Storage for indexes identifying active rows.

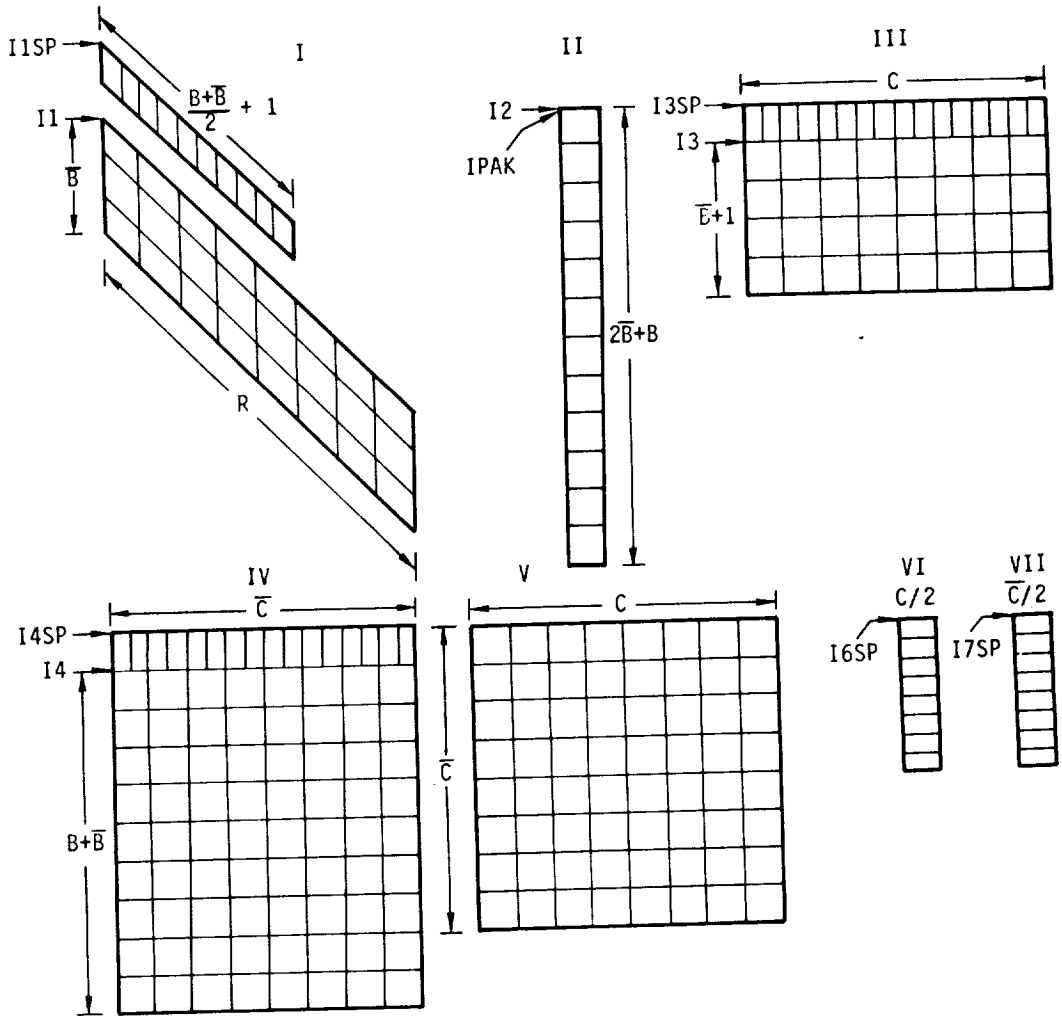


Figure 1. Definition of storage areas for DECØ1P.

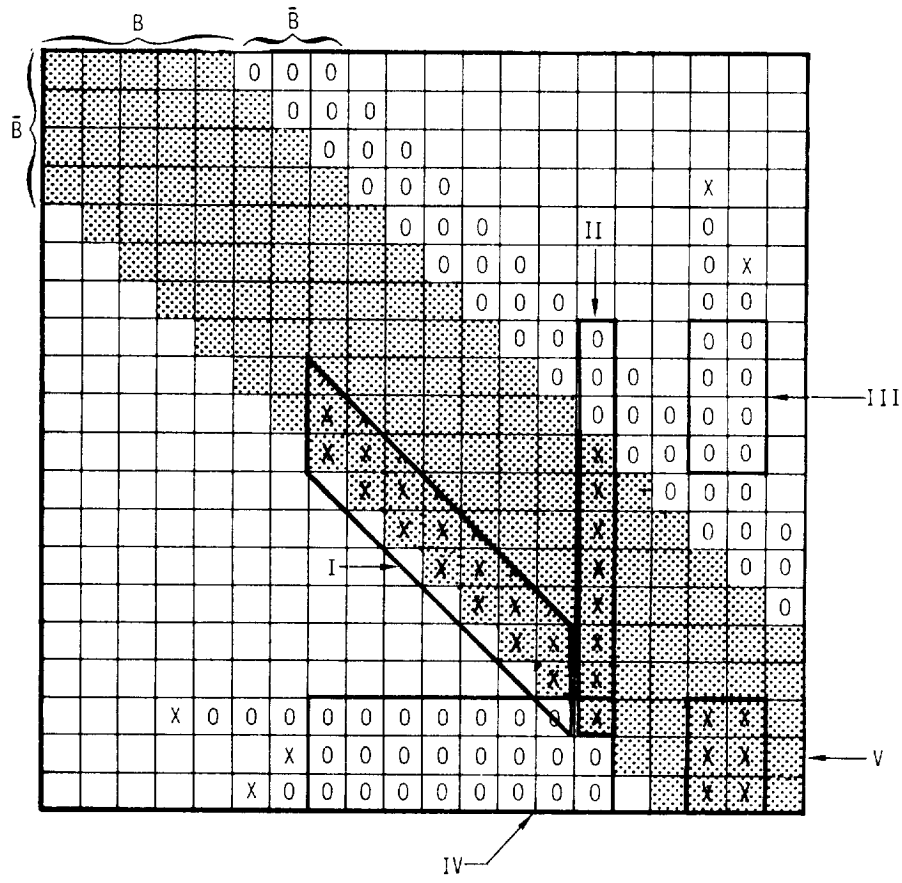


Figure 2. Location of storage areas within the matrix

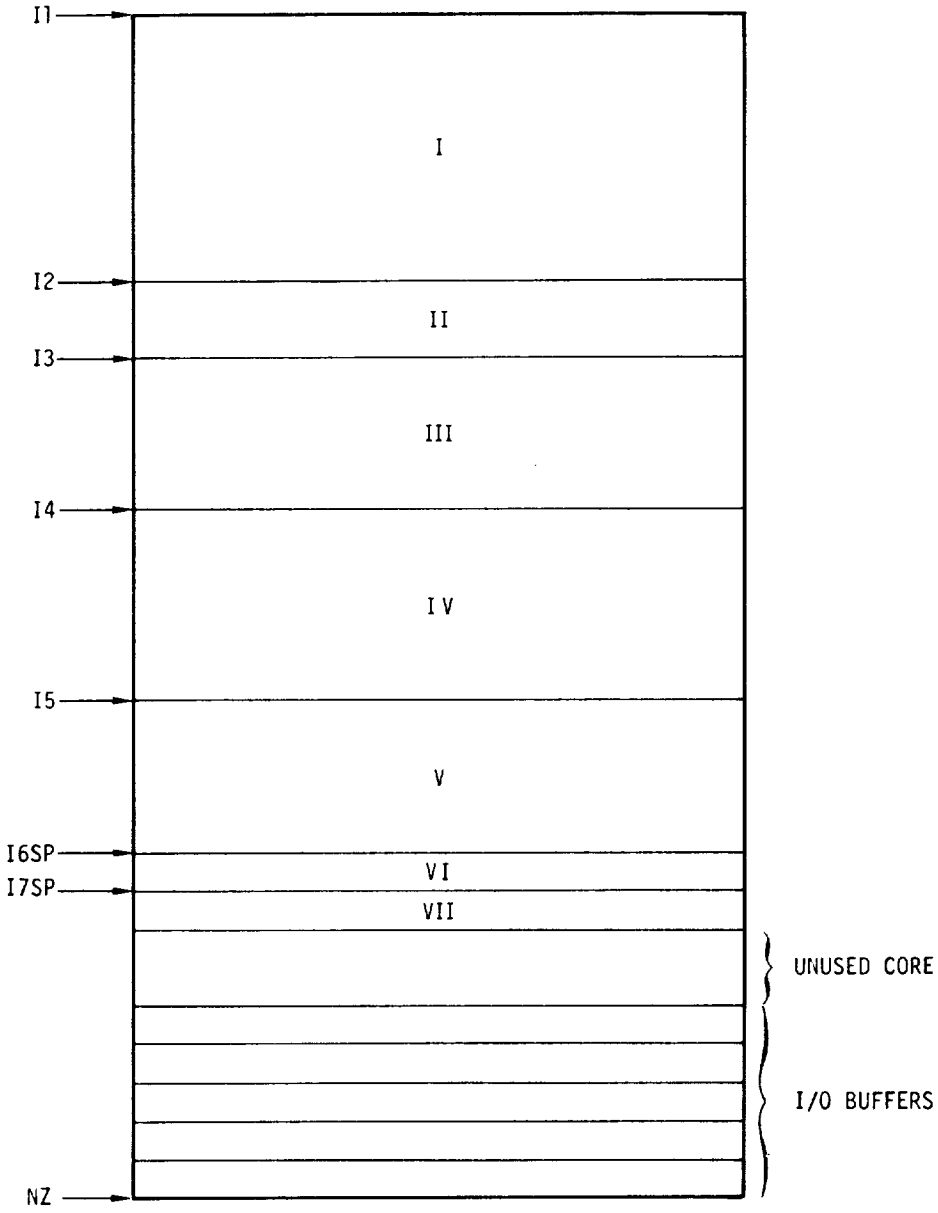


Figure 3. Allocation of core for DECØMP

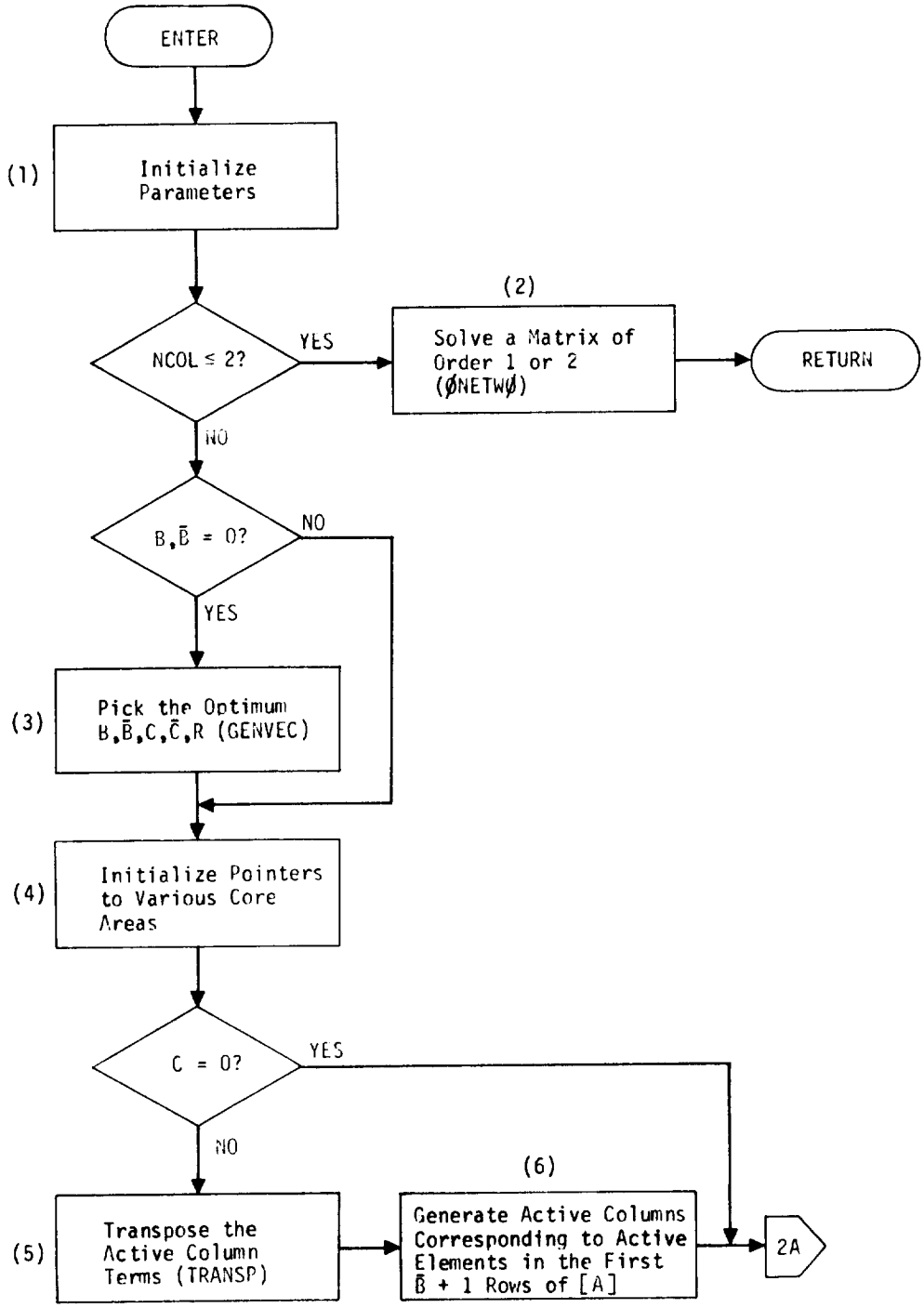


Figure 4.(a) DECØMP program flow

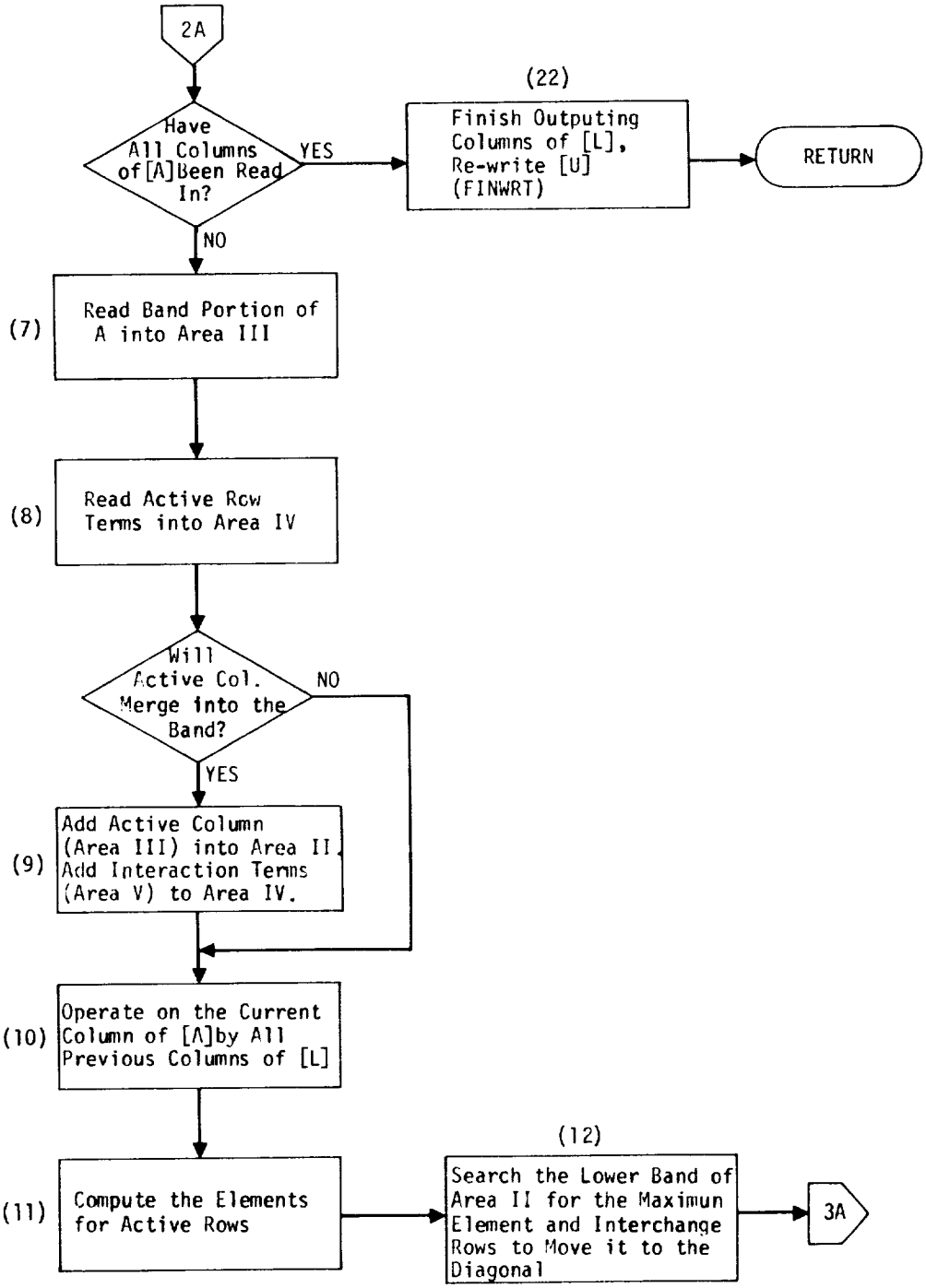


Figure4.(b) DECØMP program flow

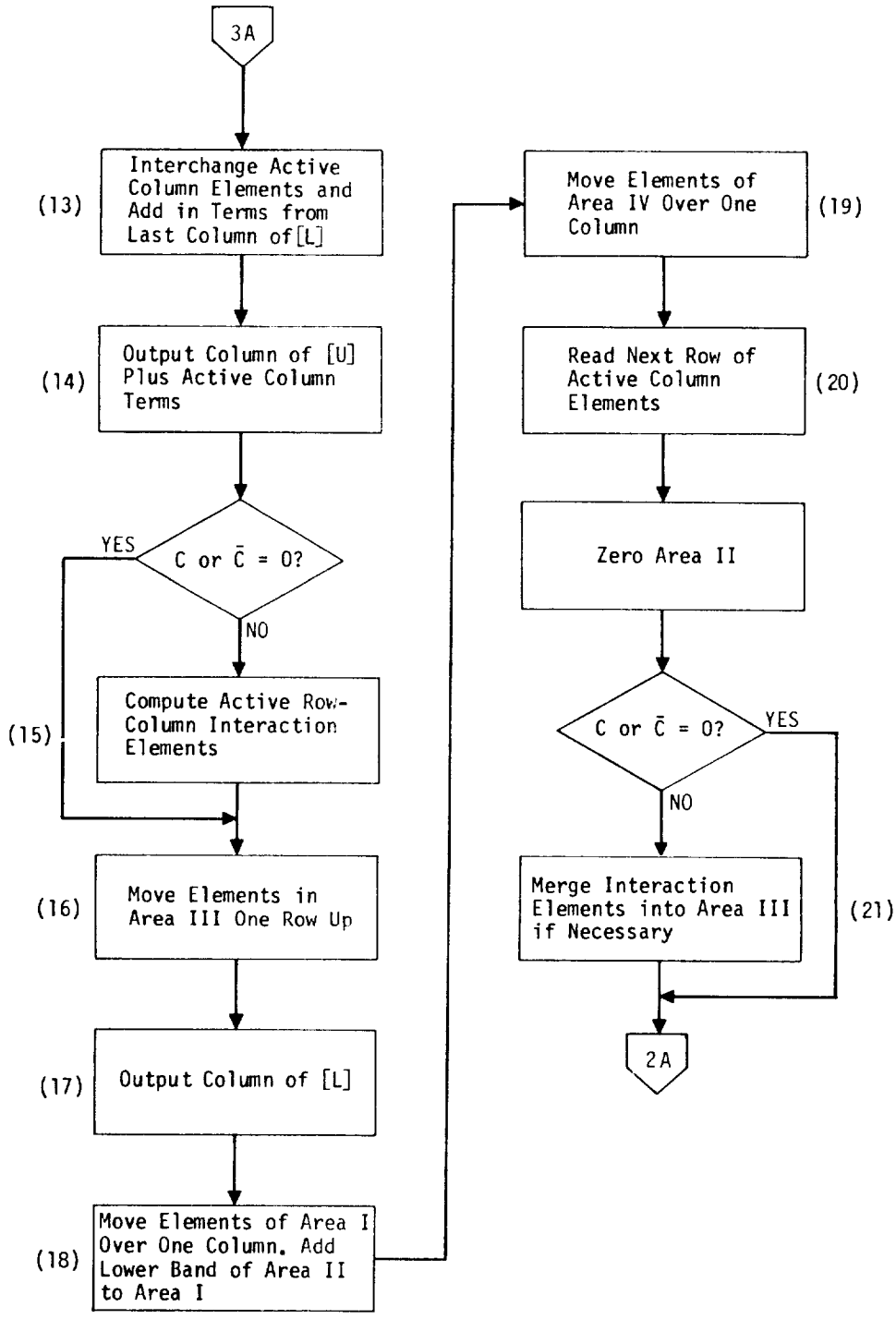


Figure 4.(c) DECØMP program flow

3. Program Flow: The flow chart in Figure 4 gives the logical program flow of DECØMP. The following comments expand on certain portions of the flow chart:

(1) Allocate buffers, initialize the determinant, and write header records.

(2) If the order of [A] is 1 or 2, subroutine ØNETWØ is called to handle the decomposition.

(3) If B and $\bar{B}$ are input as zero, GENVEC is called to pick the optimum parameters for decomposition.

(4) The pointers into the various areas of core shown in Figure 1 are computed.

(5) If there exist active elements in the upper triangle, TRANSP is called to transpose these elements.

(6) Active columns are initialized for all columns having an active element within the first $\bar{B}$ + 1 rows of [A].

(7) The band portion of the next column of [A] is read into Area II.

(8) Any active elements occurring below the diagonal in the current column are added into existing active rows, or new active rows are created.

(9) When an active column merges into the band, a column from Area III is added into corresponding positions in Area II, and a column of interaction elements in Area V is added to the active row terms in Area IV.

(10) The current column of [A] in Area II is operated on by the columns of [L] stored in Area I.

(11) Terms corresponding to active rows not yet merged into the band are added into Area II.

(12) The lower band portion of Area II is searched for the maximum element. Rows are interchanged to bring it to the diagonal position, and the interchange index is stored in Area I.

(13) Active columns elements are interchanged corresponding to the interchange within the band. If a column of [L] is about to be output (i.e., B + $\bar{B}$ columns of [L] are stored in Area I), terms arising from that column are added into the active columns.

(14) The column of [U] from Area II plus a row of active column terms in Area III are output.

(15) If active rows and columns exist, terms arising from their interaction are computed and added into Area V.

(16) Elements in the active columns are moved up one row position in Area III to replace the output elements, and make room for a new row.

(17) The first column of [L] in Area I is written out if Area I is full. The active row elements belonging to that column are also output.

(18) The columns of [L] in Area I are moved over one column and the lower band portion of Area II is stored in Area I.

(19) Active rows in Area IV are moved over one column.

(20) The next row of active elements in the upper triangle of [A] is read. Elements are added to existing active columns, or new columns are created.

(21) When possible, a row of interaction elements in Area V is merged into the bottom row of active column elements (Area III).

(22) After processing all columns of [A], FINWRT is called to complete the outputting of [L] and to re-write [U].

3.5.15.5 Auxiliary Subroutine TRANSP

1. Entry Point: TRANSP

2. Purpose: To do an in-core transpose of the active elements occurring outside the band and in the upper half of the matrix (i.e., transpose all elements a_{ij} such that $j - i \geq B$).

3. Calling Sequence: CALL TRANSP (X,X,NX,A,B,SCRFIL)

X - An area of core available to TRANSP.

NX - Number of words available at X.

A - GINØ file containing the input matrix [A] - integer.

B - Upper bandwidth of matrix [A].

SCRFIL - GINØ file where transposed elements are stored.

4. Method: The input matrix [A] is read, and all elements occurring outside the band in the upper triangle are stored in core, along with their row and column position. This list is then searched and elements output in transposed order.

3.5.15.6 Auxiliary Routine ØNETWØ

1. Entry Points: ØNETWØ,FINWRT

2. Purpose: ØNETWØ is a separate routine whose sole responsibility is to solve matrices of order one or two.

FINWRT is a section of code separated from DECØMP due to compiler overflow. Its function is to finish outputting the remaining columns of [L] and to re-write the columns of [U].

3.5.15.7 Auxiliary Subroutine GENVEC

1. Entry Point: GENVEC

2. Purpose: To pick the optimum B, $\bar{B}$, C, $\bar{C}$, and R for a given matrix [A].

3. Calling Sequence: CALL GENVEC ($\$n_1$,BUF,A,NX,X,N,B,BBAR,C,CBAR,R,IENTRY)

n_1 - Statement to which control is transferred if a null column is discovered in [A].

BUF - Location of a GINØ buffer.

A - GINØ file containing the input matrix [A].

X - An area of core available to GENVEC.

NX - Number of words of core available at X.

N - The order of the matrix [A].

B,BBAR, C,CBAR, R - Integer output parameters giving the optimum values for the upper and lower bandwidths, the number of active rows and columns, and the number of columns of [L] held in core.

IENTRY - = 1 implies DECØMP called GENVEC.
= 2 implies CDCØMP called GENVEC.

4. Method: The following logic flow gives the means by which the optimum bandwidths are chosen.

A. Locate extreme non-zero terms

1. Initialize active column list to zero. The length of the list is equal to the maximum value of the upper bandwidth (B) that is of interest. ($B_{max} = \frac{10^5}{NM_B}$, or the order of the problem, whichever is less, where N is the order of the matrix and M_B is the arithmetic time in μ seconds for performing one multiply and one add.

2. Initialize the column list to zero. The length of the list is the order of the problem.

3. Locate the non-zero elements in the next column of the matrix.

4. Insert the column number of all non-zero elements in the correct row position of the column number list with the following constraints:

   a. Consider only elements in the lower triangle.

   b. Do not insert column numbers in row positions already occupied.

5. Insert the row number of the non-zero element located in the lowest numbered row into the column position of the row number list for the current column under consideration.

6. Return to step 4 until all columns of the matrix have been processed.

B. Determine active columns

1. Zero counter E

   Set Counter F to N-1

   Set Counter G to N-1

   Set Counter H to 2

2. Beginning with the last entry in the row number list, subtract the current value of counter H, and test for a negative sign. If negative, increment counter E by one.

3. The current value of counter E is the number of active columns when the upper bandwidth is equal to the current value of counter F. Compare counter E with the existing entry in the active column list for the bandwidth indicated by counter F. Update the active column list if counter E is greater than the existing entry.

4. Decrement counter F by one. Return to step 2 unless counter F is zero.

5. Decrement counter G by one. Increment counter H by one. Set counter F to the value of counter G. Zero counter E. Return to step 2 unless counter G is zero.

6. The final active column list contains the maximum number of active columns for bandwidths of unity through the maximum upper bandwidth investigated.

7. Prepare reduced active column list by extracting pairs - minimum B and associated C for unique values of C.

C. Determine active rows for given B and $\bar{B}$

1. Extract pairs (row number, L, and column number, K) from the column number list for which

$$L - K > \bar{B}$$

2. Consider a new list of pairs consisting of K and L + B. For each pair (K, L + B) determine the number of remaining pairs $(K_i, (L + B)_i)$ for which

$$K_i < L + B \quad \text{and} \quad (L + B)_i \geq K$$

3. The maximum number of pairs satisfying the relation in step 2 for any single pair is $\bar{C}$ for the given B and $\bar{B}$.

D. Select B, B, C and $\bar{C}$ for Minimum Decomposition Time

1. Select the next value of B and associated C from the reduced active column list (begin with maximum B of interest).

2. Assume $B = \bar{B}$ and $C = \bar{C}$.

3. Calculate R and T with preliminary timing equations.

4. Save the minimum T and the associated B and C.

5. Return to step 1 unless all entries in reduced active column list have been used.

6. If the matrix is unsymmetric use B and C from step 4 and set $\bar{B} = B$.

7. Determine C for a given $\bar{B}$ and B.

8. Calculate R and T with the preliminary timing equations.

9. Compare with the previous minimum T.

10. If the new T exceeds the previous minimum by more than 20% or $\bar{B}$ = maximum B of interest go to 12.

11. Save the minimum T along with associated B, $\bar{B}$, C and $\bar{C}$. Increment $\bar{B}$ by 2% of the B associated with minimum T and go to 7.

12. Return to B = $\bar{B}$.

13. Decrement $\bar{B}$ by 2% of the B associated with minimum T and determine the associated C.

14. Calculate R and T with the preliminary timing equations.

15. Compare with the previous minimum T while decrementing $\bar{B}$.

16. If the new T exceeds the previous minimum by more than 20% or $\bar{B}$ = 2 go to 18.

17. Save the minimum T along with associated B, $\bar{B}$, C and $\bar{C}$ and go to 13.

18. Save values of B, $\bar{B}$, C and $\bar{C}$ associated with the minimum value of T from upsearch and downsearch on $\bar{B}$, for use in decomposition.

3.5.15.9 Auxiliary Function FINDC

1. Entry Point: FINDC

2. Purpose: To find the number of active rows ($\bar{C}$) for a given B and $\bar{B}$.

3. Calling Sequence: CALL FINDC (B,BBAR,N,X,Y,CBAR)

   B - Upper bandwidth - integer.

   BBAR - Lower bandwidth - integer.

   N - Order of the problem.

   X - Column number list.

   Y - Scratch vector.

   CBAR - The number of active rows, $\bar{C}$.

4. Method: See step C in the GENVEC method.

3.5.15.10 Auxiliary Routine TIMEEQ

1. Entry Points: T,TFIN,RCØRE

2. Purpose: To compute the preliminary and final timing equations for decomposition, and to compute the core allocation function.

3. Calling Sequences: CALL T (B,BBAR,C,CBAR,R,IENTRY,N,TIM)

CALL TFIN (B,BBAR,C,CBAR,R,IENTRY,N,TIMEX)

CALL RCØRE (B,BBAR,C,CBAR,N,IENTRY,NX,R)

B,BBAR
C,CBAR, - Upper and lower bandwidths, number of active rows and columns.

IENTRY - {=1 implies entry was from DECØMP.
{=2 implies entry was from CDCØMP.

N - The order of the problem.

TIM - Floating point value for the preliminary time.

TIMEX - Floating point value for the final timing equation.

NX - Number of core words available to DECØMP or CDCØMP.

R - The number of columns of [L] that can be held in core (output by RCØRE, input to T and TFIN).

4. Method: The following equations are evaluated to give the desired output.

A. Core Function

$$R = (NX - ((B + \bar{B} + 1) + 2*IENTRY*MINO(N,B + \bar{B} + \bar{B}) + 2*IENTRY*C*(\bar{B} + 2) + 2*\bar{C}*IENTRY*(MINO(B + \bar{B},N) + 1) + 2*IENTRY*C*\bar{C} + C + \bar{C}*IENTRY + \bar{C}) - 6*SYSBUF)/(2*IENTRY*\bar{B}) \quad (4)$$

B. Preliminary Timing Equation

$$TIM = \frac{N}{10^6} [M_B\bar{B}R + M_C(\bar{B}C + \bar{B}\bar{C} + B\bar{C} + 2C\bar{C}) + I\bar{B}(B + \bar{B} - R - 1)] \quad (5)$$

where

M_B = Arithmetic time in μ seconds for one term inside the band.

M_C = Arithmetic time in μ seconds for one term in the active row or active column.

I = I/Ø time in μ seconds for one term.

C. Final Timing Equations

TIMEX is a function of T_1, T_2, T_3 and T_4 as defined below (P = matrix packing time in μ seconds for one term).

T_1 is given by:

$$T_1 = K_1 [M_B \bar{B} R + I\bar{B}(B + \bar{B} - R) + P(B + 2\bar{B})] , \tag{6}$$

where

$$K_1 = \begin{cases} N - B - 2\bar{B} , & \text{if } N - B - 2\bar{B} > 0 \\ 0 , & \text{if } N - B - 2\bar{B} \leq 0 . \end{cases} \tag{7}$$

T_2 is given by:

$$T_2 = \frac{K_2}{2} [\bar{B} K_2 M_B + (K_3 - R)(I - M_B)\bar{B} + 2P\bar{B} + PK_2] , \tag{8}$$

where

$$K_2 = K_3 = B + \bar{B} \tag{9}$$

if $N \geq B + 2\bar{B}$; otherwise,

$$K_2 = N - B , \tag{10}$$

and

$$K_3 = \begin{cases} B + \bar{B} & \text{if } N \geq B + \bar{B} \\ N & \text{if } N < B + \bar{B} . \end{cases} \tag{11}$$

T_3 is given by:

$$T_3 = \frac{\bar{B}^3}{3} M_B + \frac{K_4^3}{2} I + P\bar{B}K_5 . \tag{12}$$

where

$$K_4 = \begin{cases} B + \bar{B} - R , & \text{if } B - R \leq 0 \\ \bar{B} , & \text{if } B - R > 0 . \end{cases} \tag{13}$$

and

$$K_5 = B + \frac{3}{2}\bar{B} , \tag{14}$$

if $N \geq B + 2\bar{B}$. Otherwise,

$$K_4 = \begin{cases} N - R , & \text{if } N - R \leq \bar{B} \\ \bar{B} , & \text{if } N - R > \bar{B} \end{cases} . \tag{15}$$

and

$$K_5 = N \tag{16}$$

T_4 is given by:

$$T_4 = (N - \bar{B})[M_C(\bar{B}C + B\bar{C} + \bar{B}\bar{C} + C\bar{C}) + P(C + \bar{C})] . \tag{17}$$

Finally,

$$TIMEX = (T_1 + T_2 + T_3 + T_4)10^{-6} , \tag{18}$$

where TIMEX is the total estimated time for decomposition.

3.5.15.11 Auxiliary Subroutine DLØØP

1. Entry Points: DLØØP,DDLØØP,XLØØP

2. Purpose: To improve efficiency of FØRTRAN generated code for several loops in DECØMP.

3.5.15.12 Design Requirements

Core storage requirements depend on the parameters B, $\bar{B}$, C and $\bar{C}$. Areas II, III, IV, and V must reside in core at all times, along with a minimum of two columns of Area I, and five GINØ buffers. GENVEC is designed to pick the combination of B, $\bar{B}$, C, and $\bar{C}$ such that the problem will allocate if at all possible.

The file containing [U] is not in standard format, as the active column terms are stored out of place. For this reason [L] and [U] should be used as input only to GFBS or an associated routine.

3.5.15.13 Information Messages

1. CØNMSG is called at entry and exit from DECØMP. The line

XXXX DECØMP

will appear twice per decomposition. The actual execution time of DECØMP will be the difference in the times (where XXXX = time in seconds).

2. 3023 B = XXX C = XXX R = XXX

3028 BBAR = XXX CBAR = XXX R = XXX

These messages give the parameters chosen for the decomposition.

3. 3027 DECØMPØSITIØN TIME ESTIMATE = XXX

Gives the estimated time for the decomposition in seconds.

3.5.15.14 Diagnostic Messages

Fatal error messages 3008 and 3025 may occur.

3.5.16 CDCØMP (Complex Matrix Decomposition)

3.5.16.1 Entry Point: CDCØMP

3.5.16.2 Purpose

To decompose a complex square matrix [A] into the form [A] = [L][U] where [L] is a unit lower triangular matrix and [U] is an upper triangular matrix.

3.5.16.3 Calling Sequence

```
CALL CDCØMP ($n1,X,X,X)
CØMMØN /CDCMPX/ A(7),L(7),U(7),SCR(3),DET(2),PØWER,NX,MINDIA,B,BBAR,C,CBAR,R
```

n_1 - Location in calling program where control is transferred if [A] is singular.

X - An area of working storage.

A - Input matrix control block for [A].

L,U - Output matrix control blocks for [L] and [U].

SCR(3) - Three scratch files available for use.

DET(2) - Two double precision words where the real and imaginary values of the determinant are stored.

PØWER - Scale factor to be applied to the determinant (det ([A]) = DET*10**PØWER).

NX - Number of computer words at X.

MINDIA - Double precision word where the modulus of the minimum diagonal element of [U] is stored.

B,BBAR, C,CBAR, R - If B = BBAR = 0, compute and store B,BBAR,C,CBAR,R before decomposing [A]. If B or BBAR ≠ 0, use previously stored values of B,BBAR,C,CBAR and R for decomposing.

3.5.16.4 Method

CDCØMP is simply a copy of DECØMP with the arithmetic statements replaced by complex arithmetic. Pointers to storage areas were modified to accommodate the extra words needed.

3.5.16.5 Auxiliary Subroutine CTRNSP

Purpose: Complex version of TRANSP (see section 3.5.15.5).

3.5.16.6 Auxiliary Subroutine CØM12

Purpose: Complex version of ØNETWØ (see section 3.5.15.6).

3.5.16.7 Design Requirements

See subroutine descriptions for DECØMP, section 3.5.15.

3.5.16.8 Diagnostic Messages

See subroutine descriptions for DECØMP, section 3.5.15.

3.5.17 FBS (Forward - Backward Substitution).

3.5.17.1 Entry Point: FBS.

3.5.17.2 Purpose

Given the decomposition of a real symmetric matrix [A] = [L] [D] $[L]^T$ FBS will perform the forward-backward pass necessary to solve the system of linear equations [A] [X] = [B].

3.5.17.3 Calling Sequence

```
CALL FBS(Z,Z)

COMMON/FBSX/L(7),U(7),B(7),X(7),NZ,PREC,SIGN
```

L,U - Matrix control blocks for the lower and upper triangular factors output from SDCOMP.

B,X - Matrix control blocks for the matrices [B] and [X].

NZ - Number of computer words at Z.

PREC - $\begin{cases} 1, \text{ perform arithmetic in single precision.} \\ 2, \text{ perform arithmetic in double precision.} \end{cases}$

SIGN - $\begin{cases} +1, \text{ solve } [A]\,[X] = [B]. \\ -1, \text{ solve } [A]\,[X] = -[B]. \end{cases}$

Z - An area of working storage.

3.5.17.4 Method

Mathematical Considerations. Given the unit upper and lower triangular matrices [L] and $[L]^T$, with the diagonal matrix [D] stored over their diagonals, FBS solves the two systems of equations given by

$$[L]\,[Y] = \pm[B]$$

and

$$[L]^T\,[X] = [D]^{-1}[Y]$$

Elements of [Y] and [X] are given by

$$y_{ij} = b_{ii} - \sum_{k=1}^{i-1} \ell_{ik} y_{kj}$$

$$x_{ij} = y_{ij}/d_i - \sum_{k=i+1}^{n} \ell_{ki} x_{ij}$$

Program Flow. Overall program flow is identical to that of GFBS (see section 3.5.19). The only difference in the two routines is FBS uses the decomposed matrices output from SDCØMP, while GFBS uses those output by DECØMP. Likewise, the computed equations differ slightly.

3.5.17.5 Design Requirements

One colunn of [B] and one GINØ buffer must fit in core.

3.5.17.6 Diagnostic Messages

See GFBS (see section 3.5.19).

3.5.18 SSG3A (Driver for FBS).

3.5.18.1 Entry Point: SSG3A.

3.5.18.2 Purpose

To solve [A] [X] = [B] using [A] = [L] $[L]^T$. On option to compute the residual matrix [RES] = [A] [X] - [B].

3.5.18.3 Calling Sequence

```
CALL SSG3A(FILEA,FILEL,FILELT,FILEB,FILEX,SCR1,SCR2,IRES,FILER)
```

FILEA - GINØ file name of [A] - integer - input.

FILEL - GINØ file name of [L] - integer - input.

FILELT - GINØ file name of $[L]^T$ - integer - input.

FILEB - GINØ file name of [B] - integer - input.

FILEX - GINØ file name of [X] - integer - input.

SCR1 - GINØ name of scratch file - integer - input.

SCR2 - GINØ name of scratch file - integer - input.

IRES - Option for residual vector - integer - input. IRES = 0 suppresses calculation of residual vector.

FILER - GINØ file name of residual vector - integer - input.

3.5.18.4 Method

/FBSX/ is set to compute [A] [X] = [B].

SSG2B is called to compute [RES] = [A] [X] - [B] (see section 3.5.17.3).

For each column of [X], $\{X_i\}$, ε_i is computed.

$$\varepsilon_i = \{X_i\}^T\{RES_i\}/\{B_i\}^T\{X_i\}$$

3.5.18.5 Design Requirements

Open core at /SSGA3/.

3.5.19 GFBS (General Forward - Backward Substitution).

3.5.19.1 Entry Point: GFBS.

3.5.19.2 Purpose

Given the decomposition of a general square matrix [A] = [L] [U], GFBS will solve the system of equations [A] [X] = [L] [U] [X] = $\pm$ [B]

3.5.19.3 Calling Sequence

CALL GFBS (Z,Z)

COMMON/GFBSX/L(7),U(7),B(7),X(7),NZ,PREC,ISIGN

L,U,B,X - Matrix control blocks for the matrices [L], [U], [B], and [X].

X(5) - Desired output type for [X].

NZ - Number of computer words available at Z.

PREC - $\begin{cases} 1, \text{ use single precision arithmetic} \\ 2, \text{ use double precision arithmetic} \end{cases}$

ISIGN - $\begin{cases} 1, \text{ solve } [A]\,[X] = \pm\,[B]. \\ -1, \text{ solve } [A]\,[X] = -[B]. \end{cases}$

Z - An area of working storage.

3.5.19.4 Method

1. Mathematical Considerations. Given [L], [U] (n by n lower and upper triangular matrices) and [B] (a n by m matrix) GFBS solves the two systems of equations

$$[L]\,[Y] = \pm\,[B]$$

and

$$[U]\,[X] = [Y]$$

Elements of [Y] and [X] are given by

$$y_{ij} = b_{ij} - \sum_{k=1}^{i-1} \ell_{ik} y_{kj} \qquad (1)$$

$$x_{ij} = [y_{ij} - \sum_{b=i+1}^{n} u_{ik} x_{kj}]/u_{ii} \qquad (2)$$

2. Initialization Phase. The type of arithmetic to be performed is computed as a function of the type of [L], [B], and the precision requested by the calling routine. Corresponding transfer vectors are set up to transfer control to the proper arithmetic computation. Core storage is filled with as many columns of [B] as possible.

3. Forward Pass. The intermediate values of [Y] are computed directly over the columns of [B] currently in core. This is a forward pass on [L] since it is read sequentially forward. Interpretation of Equation 1 shows that y_{1j} is complete after the first column of [L] has been read, y_{2j} after column 2, etc. Elements of [L] are read one at a time via INTPK and the appropriate term subtracted off.

4. Backward Pass. Elements of [X] are computed by processing [U] in the reverse order. Reading the last column of [U] completes $x_{n,j}$, the n-1 column of [U] gives $x_{n-1,j}$, etc. In order to facilitate the reading of [U], it was written in the reverse order by DECØMP, allowing a forward pass to be made in actuality.

5. Output Phase. The finished columns of [X] are packed and output via PACK. If more columns of [B] still exist, core is once again filled with vectors of [B] and the process repeated until all columns of [X] have been computed.

3.5.19.5 Design Requirements

At least one column of [B] must be unpacked in core, along with one GINØ buffer.

3.5.19.6 Diagnostic Messages

If insufficient core is available for GFBS, fatal message 3008 occurs.

If a diagonal term does not exist for a column of [U], fatal message 3005 occurs. This is noramlly detected in DECØMP implying care was not taken in processing singular matrices in the calling routine calling DECØMP.

3.5.20 SØLVER (Simultaneous Equation Solution Routine).

3.5.20.1 Entry Point: SØLVER.

3.5.20.2 Purpose

To perform the following three functions:

1. Solve [A] [X] = -[B] for X where A is a real symmetric matrix which has been decomposed by SDCØMP.
2. Evaluate the matrix equation $[E] = [D] + [B]^T [X]$
3. On option, compute

$$\varepsilon = \frac{||[E]||}{||[D]||}$$

3.5.20.3 Calling Sequence

CALL SØLVER(L,U,X,B,D,E,EPS,FLAG,SCR)

where:

L,U are the GINØ file names of the data blocks containing the lower and upper triangular factors of A, respectively.

X is the GINØ file name of the data block where the solution matrix will be written.

B is the GINØ file name of the data block containing the right hand matrix in [A] [X] = -[B]

D,E are the GINØ file names of the data blocks in Equation (2) above.

EPS is the real single precision result of the division in Equation (3).

SCR is the GINØ file name of a scratch file for use by SØLVER.

FLAG $\begin{cases} \neq 0, \text{ compute } \varepsilon \text{ from Equation (3) above and store in EPS} \\ = 0, \text{ do not compute } \varepsilon \end{cases}$

/SØLVRX/ is open core for SØLVER

Note:

1. L must be output from subroutine SDCØMP.
2. If D = A, then E is the error residual matrix.

3.5.20.4 Method

/FBSX/ is initialized and FBS is called to solve [A] [X] = -[B].

/MPYADX/ is initialized and MPYAD is called to compute [E] = [D] + $[B]^T$ [A]. FLAG is tested. If FLAG = 0, return is made. Otherwise, INTPK is called to unpack and interpret each non-zero term of E. The cumulative sum of the absolute value of all non-zero terms of E is formed in double precision. This operation is then performed on D. Finally ε is the division of these two quantities and is stored in EPS in single precision.

3.5.20.5 Design Requirements

/SØLVRX/ must be inserted at the end of the overlay segment containing SØLVER.

3.5.20.6 Diagnostic Messages

The following system fatal message may be issued by SØLVER:

3001

3.5.21 DMPY (Multiply a Diagonal Matrix by an Arbitrary Matrix).

3.5.21.1 Entry Point: DMPY.

3.5.21.2 Purpose

To pre or post multiply an abritrary matrix [B] by a diagonal matrix, i.e. [C] = [D][B] or [C] = [B][D].

3.5.21.3 Calling Sequence

```
CALL DMPY(Z,Z)

COMMON/DMPYX/D(7),B(7),C(7),NZ,FLAG,SIGN
```

Z - An area of working storage.

NZ - The number of computer words at Z.

D - Matrix control block for the diagonal matrix.

B - Matrix control block for the general matrix.

C - Matrix control block for the product matrix.

C(1) must contain the GINO file name prior to entry.

C(5) must contain the arithmetic type of the elements of C.

DMPY will accumulate C(2) and C(6) and set C(7) = 0.

C(3) and C(4) may be set by the user before or after the call.

FLAG - $\begin{cases} = 0, \text{ pre-multiply B by D.} \\ \neq 0, \text{ post-multiply B by D.} \end{cases}$

SIGN - $\begin{cases} +1, \text{ form positive product.} \\ -1, \text{ form negative product.} \end{cases}$

3.5.21.4 Method

The type of arithmetic is determined (real or complex). The elements of the diagonal matrix, D, are unpacked at Z. INTPK is used to read and interpret the non-zero elements of B, column by column. The following equations are used:

Pre-multiplication: $c_{ij} = d_i b_{ij}$

Post-multiplication: $c_{ij} = b_{ij} d_j$

3.5.21.5 Design Requirements

Double precision arithmetic is used throughout.

The amount of core at Z must be sufficient to hold the unpacked diagonal terms of D and two GINØ buffers.

The dimensions of D and B must be compatible although this is not checked.

3.5.21.6 Diagnostic Messages

The following system fatal messages may be issued by DMPY:

3001

3002

3.5.22 ELIM (Perform a Matrix Reduction).

3.5.22.1 Entry Point: ELIM.

3.5.22.2 Purpose

To perform a matrix reduction on the structural model by computing the matrix equation:

$$[K_{ii}] = [\overline{K}_{ii}] + [\overline{K}_{ji}]^T[G_j] + [G_j]^T[\overline{K}_{ji}] + [G_j]^T[\overline{K}_{jj}][G_j]$$

3.5.22.3 Calling Sequence

CALL ELIM(KIIB,KJIB,KJJB,GJ,KII,SCR1,SCR2,SCR3)

where KIIB, KJIB, KJJB, GJ and KII are the GINØ file names of each of the matrices in the above equation. SCR1, SCR2 and SCR3 are GINØ file names of three scratch files used by ELIM.

Open core for ELIM is defined by /ELIMX/.

3.5.22.4 Method

ELIM computes the above matrix equation by three successive calls to MPYAD. These are as follows:

$$[SCR1] = [\overline{K}_{jj}][G_j] + [\overline{K}_{ji}]$$

$$[SCR2] = [\overline{K}_{ji}]^T[G_j] + [\overline{K}_{ii}]$$

$$[K_{ii}] = [G_j]^T[SCR1] + [SCR2]$$

3.5.22.5 Design Requirements

/ELIMX/ must be included at the end of the segment in which ELIM resides.

3.5.23 FACTØR (Decompose a Matrix Into Triangular Factors).

3.5.23.1 Entry Point: FACTØR.

3.5.23.2 Purpose

To decompose a symmetric matrix into its two triangular factors.

3.5.23.3 Calling Sequence

CALL FACTØR (A,L,U,SCR1,SCR2)

where A, L, U are the GINØ file names of the data blocks for the symmetric matrix to be decomposed, its lower triangular factor and its upper triangular factor, respectively. If U is negative, FACTØR will decompose the symmetric matrix [A] on file A such that

$$[A] = [C][C]^T ,$$

where the Cholesky decomposition is used. [C] will be written on U as a lower triangular matrix. L will contain the lower triangular factor of a standard non-Cholesky decomposition. SCR1 and SCR2 are GINØ file names of two scratch files for use by FACTØR. The common block /FACTRX/ is open core for FACTØR.

3.5.23.4 Method

FACTØR initializes /SFACT/ and calls SDCØMP to accomplish the decomposition.

3.5.23.5 Design Requirements

/FACTRX/ must be included at the end of the segment which contains FACTØR.

3.5.23.6 Diagnostic Messages

The following system fatal message may be issued by FACTØR:

3005

3.5.24 TRANP1 (Driver for TRNSP).

3.5.24.1 Entry Point: TRANP1.

3.5.24.2 Purpose

To drive TRNSP to compute $[B] = [A]^T$.

3.5.24.3 Calling Sequence

CALL TRANP1(FILEA,FILAT,NSCRTH,SCR1,SCR2,SCR3,SCR4,SCR5,SCR6,SCR7,SCR8).

FILEA - GINØ name of [A] - integer - input.

FILAT - GINØ name of $[A]^T$ - integer - input.

NSCRTH- Number of scratch files - integer - input.

SCR1, ... , SCR8 - GINØ names of scratch files - integer - input.

3.5.24.4 Method

/TRNSPX/ is set based on the trailer of FILEA.

3.5.24.5 Design Requirements

Open core at /DTRANX/.

3.5.25 TRNSP (Matrix Transpose).

3.5.25.1 Entry Point: TRNSP.

3.5.25.2 Purpose

To compute $[A]^T$ given [A].

3.5.25.3 Calling Sequence

CALL TRNSP(Z)

Z - Array of core.

COMMON/TRNSPX/MCBA(7),MCBAT(7),LORE,NSCRTH,SCR(8)

MCBA - Matrix control block for [A] - input.

MCBAT - Matrix control block for $[A]^T$ - input.

LORE - Length of Z array - integer - input.

NSCRTH - Number of scratch files available - integer - input. $1 \leq NSCRTH \leq 8$.

SCR - List of GINO names of scratch files - integer - input.

3.5.25.4 Method

Three methods are possible:

Method 1 - In-core matrices.

If the full matrix can be held in core, [A] is unpacked into core, and packed out onto $[A]^T$.

Method 2 - Simple sub-matrices.

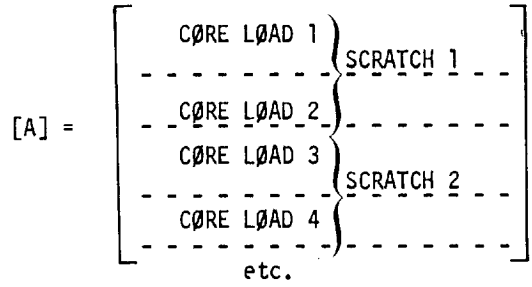
The matrix [A] is broken up one column at a time into submatrices. The submatrices are written on scratch files, read in and transposed one at a time. The break-up of [A] can be pictured as follows:

$$[A] = \begin{bmatrix} \text{SCRATCH 1} \\ \hline \text{SCRATCH 2} \\ \hline \text{SCRATCH 3} \\ \hline \text{etc.} \end{bmatrix}$$

If insufficient scratch files are available to hold [A], multiple passes may be made

Method 3 - Multiple passes on submatrices.

If multiple passes on [A] are necessary in Method 2, it may be more efficient to create larger submatrices and pass each submatrix several times rather than several passes on [A]. The break-up of [A] can be pictured as follows:



TRNSP will choose between the methods to minimize running time.

3.5.25.5 Design Requirements

Methods 2 and 3 require at least one scratch file.

Neither [A] nor $[A]^T$ may be purged.

One unpacked column of [A] and NSCRTH+1 GINØ buffers must fit into core.

3.5.25.6 Diagnostic Messages

A message indicating insufficient core is produced if the third of the above requirements is not satisfied. Either reduce the number of scratch files or increase the open core available to TRNSP.

4.1 GENERAL COMMENTS AND INDEXES

The NASTRAN modules (a module is a logical group of subroutines) documented in this section have been classified into 7 categories: 1) Executive Preface modules, 2) Executive modules, 3) Executive DMAP instructions, 4) Executive DMAP modules, 5) functional modules, 6) output modules, and 7) matrix modules.

Executive Preface modules are those which are executed prior to the execution of the first module in a DMAP sequence. They consist of: 1) XCSA (Executive Control Section Analysis), which processes the NASTRAN Executive Control Deck; 2) IFP1 (Input File Processor, Part 1), which processes the NASTRAN Case Control Deck; 3) XSØRT (Executive Bulk Data Card Sort), which sorts the NASTRAN Bulk Data Deck; 4) IFP (Input File Processor), which processes the sorted Bulk Data Deck; 5) IFP3 (Input File Processor 3), which processes bulk data cards unique to an axisymmetric conical shell problem; 6) XGPI (Executive General Problem Initialization), the heart of the Preface, which a) translates (compiles) a DMAP sequence into an internal form, the ØSCAR, - see section 2.4.2.1, and b) for problem restarts, initializes data blocks and labeled common blocks; and 7) UMFEDIT (User Master File Editor), which creates and manipulates User Master Files.

The only module classified as an Executive module, per se, is XSFA (Executive Segment File Allocator), which is the "administrative manager" of files for NASTRAN.

Executive DMAP instructions documented in this section are REPT, JUMP, CØND, EXIT and END. The DMAP instructions BEGIN, LABEL and FILE are not allocated a separate section, and therefore brief explanations follow.

1. The BEGIN DMAP instruction is a declarative DMAP instruction which denotes the beginning of a DMAP sequence. It is analogous to a computer operating system control card which calls a compiler of the system.

2. The FILE DMAP instruction is a declarative DMAP instruction which alters the normal attributes of a data block in an ØSCAR entry (see an explanation of the attributes AP, LTU, TP, NTU of a data block in the Data Section Format for functional modules section in the description of the ØSCAR, section 2.4.2.1). These attributes of a data block are used by the Executive Segment File Allocator (XSFA) module in performing its task.

3. The LABEL DMAP instruction is used to label a location in a DMAP sequence so that the location may be referenced by the DMAP instructions JUMP, CØND and REPT.

A more detailed description of these three Executive DMAP instructions can be found in section 5 of the User's Manual.

Executive DMAP modules consist of CHKPNT, SAVE, PURGE, EQUIV, PARAM and SETVAL. In addition to the descriptions in this section, the reader is referred to section 5 of the User's Manual, where further explanations of the uses of EQUIV, PURGE and CHKPNT can be found.

Functional modules comprise the bulk of the descriptions in this section. Functional modules perform the actual structural problem solution. The reader is referred to section 5 of the User's Manual for a) general comments on DMAP rules and b) the syntactical rules of the DMAP calling sequences referring to functional modules.

Output modules are those whose entire output is directed a) to the system output file and/or b) to a tape which will drive a plotting device.

Matrix modules are those which, although no different operationally from functional modules, are most likely to be used by the program user who wishes to take advantage of the Direct Matrix Abstraction capabilities of NASTRAN; therefore, they are separately categorized.

4.1.1 Use of Module Functional Descriptions

Each module documented by means of a Module Functional Description (MFD) has been assigned an integer i, and its MFD is documented in section 4.i. For functional modules, a consistent numbering scheme has been followed, wherever possible, in the MFD's. For a functional module whose assigned integer is i, then,

4.i Title

4.i.1 Entry Point

4.i.2 Purpose

4.i.3 DMAP Calling Sequence

4.i.4 Input Data Blocks

4.i.5 Output Data Blocks

4.i.6 Parameters

4.i.7 Method

4.i.8 Subroutines

4.i.9 Design Requirements

4.i.10 Diagnostic Messages

comprises this numbering scheme. The title: a) classifies the module into one of the seven categories defined in the first paragraph of section 4.1; b) defines the module name, which, if it is a DMAP module (one which is called by a DMAP instruction), is the name by which it must be called in the DMAP calling sequence; and c) defines, parenthetically, the phrase from which the name was derived. Comments on the remaining sections follow.

1. Entry Point

This section defines the entry point of the module. A module's entry point usually agrees with the module name, but there are exceptions. For example, the READ (Real Eigenvalue Analysis Displacement) module has the entry point REIG (the entry point READ is a subroutine in the GINØ collection of routines).

2. Purpose

A brief description of the purpose of the module is given. The casual or first-time reader will perhaps go no further than read the purpose.

3. DMAP Calling Sequence

The DMAP calling sequence as it appears in a Rigid Format is given (see section 3 of the User's Manual for a detailed description of the Rigid Formats in NASTRAN). DMAP calling sequences for Executive DMAP instructions and for Executive DMAP modules follow no fixed format. Refer to the individual Module Functional Descriptions for details on their DMAP calling sequences. Functional modules, which are always "called" via a DMAP calling sequence, do have a fixed format. Consider the following DMAP calling sequence for functional module SMA2, which generates the mass matrix, $[M_{gg}]$, and the damping matrix, $[B_{gg}]$:

```
SMA2   CSTM,MPT,ECPT,GPCT,DIT/MGG,BGG/V,Y,WTMASS = 1.0/V,N,NØMGG/V,N,NØBGG/V,Y,
       CØUPBAR = -1  $
```

SMA2 is the module name. The name of a module must begin with an alphabetic character

followed by up to seven additional alphanumeric characters. Following the name is a blank field. Following this blank field is the list {CSTM,MPT,ECPT,GPCT,DIT} of data blocks input to the module. The list is terminated by a slash (/). Each item in this list is separated by a comma. Note that the number of commas for this list is one less than the number of input data blocks. The second slash terminates the list {MGG,BGG} of data blocks output from the module. The rule for naming input and output data blocks is the same as for module names. Each subsequent slash terminates a parameter field. Each parameter field contains three parts separated by commas. The first part is either the letter "V" or the letter "C", defining the parameter as a variable or as a constant respectively. The second part is either the letter "Y" or the letter "N". "Y" implies "yes" the value of the parameter may be specified on a PARAM bulk data card, and "N" implies "no" the value of the parameter may not be specified on a PARAM bulk data card. The third part may be either: (a) the name of the parameter, (b) the value of the parameter, or (c) the name and the value of the parameter. A variable parameter must have a name, hence a variable parameter may not be specified only by its value. The rule governing the names of parameters is the same as that for module names. The value of a parameter may be complex double precision, complex single precision, double precision, real, integer or BCD. In the example given, the name of the first parameter is WTMASS, and its initial value (which can be overridden by a value on a PARAM card because of "Y" prior to the name) is 1.0. Note that the slash terminating the last parameter field is omitted. Although one can terminate the last parameter field with a slash, this final slash is usually omitted. A dollar sign, "$", terminates a DMAP statement.

4. Input Data Blocks

A short description of each of the module's input data blocks is given along with notes explaining what the module's design requires about the status (purged or not purged) of the data blocks. Detailed data block descriptions are found in section 2 of the Programmer's Manual.

5. Output Data Blocks

A short description of each of the module's output data blocks and an explanation of the action taken when an output data block has been pre-purged are given in this section. An output data block is said to be pre-purged if the data block has been

explicitly purged in a previous PURGE DMAP instruction, or if the data block does not appear in the DMAP calling sequence for the module.

6. Parameters

The order of DMAP parameters in a DMAP calling sequence is the same as the order of the FØRTRAN variables corresponding to the parameters in blank common at module execution time. Each variable DMAP parameter is defined as whether a) it is input data into, or output data from, the module, or both (e.g., a DMAP loop counter which is incremented within the module); b) the type of the parameter: integer, real, double precision, complex single precision, complex double precision, or BCD; and c) the default value of the parameter as defined either i) in the Module Properties List (MPL) Executive table, ii) by means of a PARAM or SETVAL DMAP instruction, or iii) by means of the DMAP statement itself. An example of the third type of default value is

```
MØDULEA    A,B,C/D,E/V,N,UVW/V,Y,XYZ=-1 $.
```

The parameter XYZ is set to -1 by the above statement. For further information on DMAP parameters see paragraph 3 above, section 2.4.2.2 in the Programmer's Manual and section 5 of the User's Manual.

7. Method

A discussion of the method used by the module writer to achieve the purpose of the module is given in this section.

8. Subroutines

The subroutines which comprise the module are described in this section. However, not all subroutines capable of being called by a module are listed here. Utility routines and matrix routines that are in the root segment are not listed in this section. These inlcude: MAPFNS, all the GINØ routines (ØPEN, WRITE, CLØSE, READ, FWDREC, BCKREC, REWIND, EØF, SKPFIL, XGINØ, GINØ, ØPNCØR), FREAD, GØPEN, WRTTRL, FNAME, CLSTAB, PRELØC, PEXIT, TMTØGØ, MESAGE, and the matrix packing and unpacking routines (BLDPK, PACK, INTPK, UNPACK). Descriptions for these routines are found in section 3.

9. Design Requirements

Design requirements peculiar to the module are presented.

10. Diagnostic Messages

Diagnostic messages unique to the module are given in this section. A detailed list of NASTRAN diagnostic messages can be found in section 6 of the User's Manual.

4.1.2 Alphabetical Index of Module Functional Descriptions

Section Number	Module Name	Section Number	Module Name
4.78	ADD	4.24	PLØT
4.56	CASE	4.23	PLTSET
4.59	CEAD	4.76	PRTMSG
4.10	CHKPNT	4.77	PRTPARM
4.13	CØND	4.16	PURGE
4.67	DDR1	4.64	RANDØM
4.68	DDR2	4.37	RBMG1
4.81	DECØMP	4.38	RBMG2
4.47	DPD	4.39	RBMG3
4.49	DSMG1	4.40	RBMG4
4.51	DSMG2	4.48	READ
4.18	END	4.11	REPT
4.17	EQUIV	4.15	SAVE
4.14	EXIT	4.35	SCE1
4.82	FBS	4.45	SDR1
4.61	FRRD	4.46	SDR2
4.58	GKAD	4.62	SDR3
4.66	GKAM	4.74	SEEMAT
4.32	GPSP	4.20	SETVAL
4.29	GPWG	4.27	SMA1
4.21	GP1	4.28	SMA2
4.22	GP2	4.30	SMA3
4.25	GP3	4.86	SMPYAD
4.31	GP4	4.36	SMP1
4.5	IFP	4.50	SMP2
4.3	IFP1	4.80	SØLVE
4.6	IFP3	4.41	SSG1
4.12	JUMP	4.42	SSG2
4.72	MATGPR	4.43	SSG3
4.71	MATPRN	4.44	SSG4
4.73	MATPRT	4.75	TABPT
4.33	MCE1	4.26	TA1
4.34	MCE2	4.65	TRD
4.84	MERGE	4.85	TRNSP
4.79	MPYAD	4.8	UMFEDIT
4.57	MTRXIN	4.60	VDR
4.70	ØFP	4.2	XCSA
4.19	PARAM	4.7	XGPI
4.83	PARTN	4.9	XSFA
4.52	PLA1	4.4	XSØRT
4.53	PLA2	4.69	XYPLØT
4.54	PLA3	4.63	XYTRAN
4.55	PLA4		

4.1.3 Alphabetical Index of Entry Points in Module Functional Descriptions

Names listed under entry point which end in the characters "BD" are block data subprograms.

Section Number	Entry Point	Module Name	Page Number
4.46.8	AI	SDR2	4.46-7
4.46.8	AJ	SDR2	4.46-7
4.46.8	AK	SDR2	4.46-7
4.46.8	AM	SDR2	4.46-7
4.46.8	AMATRX	SDR2	4.46-7
4.48.8.25	ARRM	READ	4.48-18
4.41.11.37	BAR	SSG1	4.41-27
4.41.11.22	BASGLB	SSG1	4.41-22
4.46.8	BINT	SDR2	4.46-7
4.28.8.16	BVISC	SMA2	4.28-8
4.56.1	CASE	CASE	4.56-1
4.59.8.14	CDETM	CEAD	4.59-12
4.59.8.15	CDETM2	CEAD	4.59-12
4.59.8.19	CDETM3	CEAD	4.59-14
4.59.8.9	CDIFBS	CEAD	4.59-8
4.59.8.18	CDTFBS	CEAD	4.59-14
4.59.1	CEAD	CEAD	4.59-1
4.59.8.1	CEAD1A	CEAD	4.59-3
4.59.8.8	CINFBS	CEAD	4.59-8
4.59.8.2	CINVPR	CEAD	4.59-3
4.59.8.3	CINVP1	CEAD	4.59-4
4.59.8.4	CINVP2	CEAD	4.59-4
4.59.8.5	CINVP3	CEAD	4.59-6
4.59.8.10	CMTIMU	CEAD	4.59-9
4.59.8.6	CNØRM	CEAD	4.59-7
4.59.8.7	CNØRM1	CEAD	4.59-7
4.23.8.3	CNSTRC	PLTSET	4.23-3
4.46.8	CØEF	SDR2	4.46-7
4.41.11.7	CØMBIN	SSG1	4.41-17
4.23.8.2	CØMECT	PLTSET	4.23-3
4.41.11.34	CØNE	SSG1	4.41-26
4.4.5.7	CRDFLG	XSØRT	4.4-5
4.41.11.19	CRØSS	SSG1	4.41-21
4.59.8.17	CSQRT	CEAD	4.59-13
4.59.8.12	CSUB	CEAD	4.59-10
4.59.8.16	CSUMM	CEAD	4.59-13
4.59.8.11	CXTRNY	CEAD	4.59-9

Section Number	Entry Point	Module Name	Page Number
4.78.1	DADD	ADD	4.78-1
4.49.8.6	DBEAM	DSMG1	4.49-6
4.49.8.10	DCØNE	DSMG1	4.49-7
4.81.1	DDCØMP	DECØMP	4.81-1
4.67.1	DDR1	DDR1	4.67-1
4.68.1	DDR2	DDR2	4.68-1
4.68.8.1	DDR1A	DDR1	4.68-4
4.68.8.2	DDR1B	DDR1	4.68-5
4.27.8.5	DETCK	SMA1	4.27-9
4.48.8.24	DETDET	READ	4.48-17
4.48.8.28	DETFBS	READ	4.48-19
4.48.8.15	DETM	READ	4.48-13
4.48.8.16	DETM1	READ	4.48-15
4.48.8.17	DETM2	READ	4.48-16
4.48.8.18	DETM3	READ	4.48-16
4.48.8.19	DETM4	READ	4.48-16
4.48.8.20	DETM5	READ	4.48-16
4.48.8.21	DETM6	READ	4.48-16
4.82.1	DFBS	FBS	4.82-1
4.41.11.9	DIRECT	SSG1	4.41-18
4.27.8.27	DKEF	SMA1	4.27-16
4.27.8.22	DKI	SMA1	4.27-13
4.27.8.25	DKINT	SMA1	4.27-14
4.27.8.26	DKJ	SMA1	4.27-15
4.27.8.31	DKJAB	SMA1	4.27-17
4.27.8.23	DKK	SMA1	4.27-14
4.27.8.24	DKM	SMA1	4.27-14
4.27.8.29	DK100	SMA1	4.27-16
4.27.8.33	DK211	SMA1	4.27-18
4.27.8.32	DK219	SMA1	4.27-18
4.27.8.28	DK89	SMA1	4.27-16
4.27.8.38	DMATRX	SMA2	4.27-21
4.28.8	DMEF	SMA2	4.28-3
4.28.8	DMI	SMA2	4.28-3
4.28.8	DMINT	SMA2	4.28-3
4.28.8	DMJ	SMA2	4.28-3
4.28.8	DMJAB	SMA2	4.28-3
4.28.8	DMK	SMA2	4.28-3
4.28.8	DMM	SMA2	4.28-3
4.79.1	DMPYAD	MPYAD	4.79-1
4.28.8	DM100	SMA2	4.28-3

Section Number	Entry Point	Module Name	Page Number
4.28.8	DM211	SMA2	4.28-3
4.28.8	DM219	SMA2	4.28-3
4.28.8	DM89	SMA2	4.28-3
4.47.1	DPD	DPD	4.47-1
4.47.8.1	DPDAA	DPD	4.47-7
4.47.9.2	DPDCBD	DPD	4.47-8
4.47.7.1	DPD1	DPD	4.47-3
4.47.7.1	DPD2	DPD	4.47-3
4.47.7.1	DPD3	DPD	4.47-3
4.47.7.1	DPD4	DPD	4.47-3
4.47.7.1	DPD5	DPD	4.47-3
4.24.1	DPLØT	PLØT	4.24-1
4.23.1	DPLTST	PLTSET	4.23-1
4.49.8.9	DQDMEM	DSMG1	4.49-7
4.24.8.6	DRAW	PLØT	4.24-7
4.49.8.5	DRØD	DSMG1	4.49-6
4.49.8.7	DSHEAR	DSMG1	4.49-6
4.49.1	DSMG1	DSMG1	4.49-1
4.51.1	DSMG2	DSMG2	4.51-1
4.49.8.1	DS1	DSMG1	4.49-5
4.49.8.2	DS1A	DMSG1	4.49-5
4.49.8.4	DS1ABD	DSMG1	4.49-6
4.49.8.3	DS1B	DSMG1	4.49-5
4.85.1	DTRANP	TRNSP	4.85-1
4.49.8.8	DTRMEM	DSMG1	4.49-7
4.27.8.35	D4K	SMA1	4.27-19
4.27.8.36	D5K	SMA1	4.27-20
4.27.8.37	D6K	SMA1	4.27-20
4.24.8.11	DVECTR	PLØT	4.24-10
4.48.8.23	EADD	READ	4.48-17
4.23.9.2	EDESBD	PLTSET	4.23-5
4.41.11.4	EDTL	SSG1	4.41-16
4.24.8.16	ELELBL	PLØT	4.24.12a
4.48.8.36	EMPCØR	READ	4.48-19c
4.41.11.2	EXTERN	SSG1	4.41-14
4.4.5.8	EXTINT	XSØRT	4.4-5
4.41.11.33	FCURL	SDR2	4.41-26
4.41.11.26	FDCSTM	SSG1	4.41-23
4.48.8.22	FDVECT	READ	4.48-17
4.41.11.28	FEDT	SSG1	4.41-24
4.46.8	FF100	SDR2	4.46-7

Section Number	Entry Point	Module Name	Page Number
4.41.11.27	FGPTT	SSG1	4.41-24
4.48.8.37	FILCØR	READ	4.48-19d
4.24.8.2	FIND	PLØT	4.24-4
4.46.8	FJAB	SDR2	4.46-7
4.41.11.18	FNDPNT	SSG1	4.41-21
4.24.8.12	FNDSET	PLØT	4.24-10
4.41.11.21	FNDSIL	SSG1	4.41-22
4.65.8.5	FØRM1	TRD	4.65-14
4.41.11.11	FPØNT	SSG1	4.41-19
4.61.1	FRRD	FRRD	4.61-1
4.61.8.1	FRRD1A	FRRD	4.61-5
4.61.8.2	FRRD1B	FRRD	4.61-6
4.61.8.3	FRRD1C	FRRD	4.61-6
4.61.8.4	FRRD1D	FRRD	4.61-6
4.61.8.5	FRRD1E	FRRD	4.61-7
4.61.8.6	FRRD1F	FRRD	4.61-7
4.46.8	F4	SDR2	4.46-7
4.46.8	F5	SDR2	4.46-7
4.46.8	F6	SDR2	4.46-7
4.46.8	F6211	SDR2	4.46-7
4.46.8	F6219	SDR2	4.46-7
4.46.8	F89	SDR2	4.46-7
4.24.8.4	GETDEF	PLØT	4.24-6
4.58.1	GKAD	GKAD	4.58-1
4.58.8.1	GKAD1A	GKAD	4.58-6
4.58.8.2	GKAD1B	GKAD	4.58-6
4.58.8.3	GKAD1C	GKAD	4.58-7
4.58.8.4	GKAD1D	GKAD	4.58-7
4.66.1	GKAM	GKAM	4.66-1
4.66.8.2	GKAM1A	GKAM	4.66-4
4.66.8.1	GKAM1B	GKAM	4.66-3
4.41.11.23	GLBBAS	SSG1	4.41-22
4.32.1	GPSP	GPSP	4.32-1
4.24.8.10	GPTLBL	PLØT	4.24-9
4.24.8.9	GPTSYM	PLØT	4.24-9
4.29.1	GPWG	GPWG	4.29-1
4.29.8.1	GPWG1A	GPWG	4.29-5
4.29.8.2	GPWG1B	GPWG	4.29-5
4.29.8.3	GPWG1C	GPWG	4.29-6
4.21.1	GP1	GP1	4.21-1
4.22.1	GP2	GP2	4.22-1
4.25.1	GP3	GP3	4.25-2

Section Number	Entry Point	Module Name	Page Number
4.25.8.3	GP3A	GP3	4.25-4
4.25.8.4	GP3B	GP3	4.25-7
4.25.8.2	GP3C	GP3	4.25-2
4.31.1	GP4	GP4	4.31-1
4.41.11.16	GRAV	SSG1	4.41-20
4.41.11.5	GRAVL1	SSG1	4.41-16
4.41.11.6	GRAVL2	SSG1	4.41-16
4.41.11.29	GRAVL3	SSG1	4.41-24
4.46.8	IFAC	SDR2	4.46-7
4.5.1	IFP	IFP	4.5-1
4.5.7.9	IFPPK,IFPPKI,IFPPKN	IFP	4.5-7
4.3.1	IFP1	IFP1	4.3-1
4.3.7.1	IFP1B	IFP1	4.3-2
4.3.7.2	IFP1C	IFP1	4.3-3
4.3.7.3	IFP1D	IFP1	4.3-4
4.3.7.4	IFP1E	IFP1	4.3-4
4.3.7.5	IFP1F	IFP1	4.3-4
4.3.7.6	IFP1G	IFP1	4.3-5
4.3.7.8	IFP1XY	IFP1	4.3-6
4.6.1	IFP3	IFP3	4.6-1
4.6.8.1	IFP3B	IFP3	4.6-15
4.5.7.8	IFS1P	IFP	4.5-6
4.5.7.8	IFS2P	IFP	4.5-6
4.5.7.8	IFS3P	IFP	4.5-6
4.5.7.8	IFS4P	IFP	4.5-6
4.5.7.1	IFX1BD	IFP	4.5-5
4.5.7.2	IFX2BD	IFP	4.5-5
4.5.7.3	IFX3BD	IFP	4.5-5
4.5.7.4	IFX4BD	IFP	4.5-6
4.5.7.5	IFX5BD	IFP	4.5-6
4.5.7.6	IFX6BD	IFP	4.5-6
4.5.7.7	IFX7BD	IFP	4.5-6
4.4.5.3	INITCØ	XSØRT	4.4-4
4.65.8.3	INITL	TRD	4.65-13
4.4.5.9	INTEXT	XSØRT	4.4-5
4.65.8.8	INTFBS	TRD	4.65-15
4.24.8.7	INTVEC	PLØT	4.24-8
4.48.8.40	INVERT	READ	4.48-19e
4.48.8.14	INVFBS	READ	4.48-12
4.48.8.6	INVPWR	READ	4.48-8
4.48.8.7	INVP1	READ	4.48-8

Section Number	Entry Point	Module Name	Page Number
4.48.8.8	INVP2	READ	4.48-9
4.48.8.9	INVP3	READ	4.48-9
4.48.8.41	INVTR	READ	4.48.19f
4.4.5.11	ISFT	XSØRT	4.4-6
4.27.8.7	KBAR	SMA1	4.27-10
4.27.8.17	KBEAM	SMA1	4.27-12
4.27.8.18	KCØNE	SMA1	4.27-13
4.27.8.16	KELAS	SMA1	4.27-12
4.27.8.30	KFAC	SMA1	4.27-17
4.27.8.9	KPANEL	SMA1	4.27-10
4.27.8.11	KQDMEM	SMA1	4.27-11
4.27.8.14	KQDPLT	SMA1	4.27-12
4.27.8.6	KRØD	SMA1	4.27-10
4.27.8.21	KTØRDR	SMA1	4.27-13
4.27.8.20	KTRAPR	SMA1	4.27-13
4.27.8.12	KTRBSC	SMA1	4.27-11
4.27.8.15	KTRIQD	SMA1	4.27-12
4.27.8.19	KTRIRG	SMA1	4.27-13
4.27.8.10	KTRMEM	SMA1	4.27-11
4.27.8.13	KTRPLT	SMA1	4.27-11
4.27.8.8	KTUBE	SMA1	4.27-10
4.2.5.2	LDi	XCSA	4.2-2
4.2.5.3	LiBD,LiABD	XCSA	4.2-2
4.41.11.8	LØADX	SSG1	4.41-17
4.46.8.36	MAGPHA	SDR2	4.46-19
4.28.8.12	MASSD	SMA2	4.28-7
4.28.8.7	MASSTQ	SMA2	4.28-5
4.72.1	MATGPR	MATGPR	4.72-1
4.71.1	MATPRN	MATPRN	4.71-1
4.65.8.6	MATVEC	TRD	4.65-14
4.28.8.8	MBAR	SMA2	4.28-6
4.28.8.17	MBEAM	SMA2	4.28-8
4.28.8.9	MCBAR	SMA2	4.28-6
4.33.1	MCE1	MCE1	4.33-1
4.33.8.1	MCE1A	MCE1	4.33-2
4.33.8.2	MCE1B	MCE1	4.33-2
4.33.7	MCE1C	MCE1	4.33-2
4.33.8.3	MCE1D	MCE1	4.33-2
4.34.1	MCE2	MCE2	4.34-1
4.28.8.11	MCØNE	SMA2	4.28-6
4.28.8.10	MCØNMX	SMA2	4.28-6
4.28.8.18	MCRØD	SMA2	4.28-8

Section Number	Entry Point	Module Name	Page Number
4.84.1	MERGEX	MERGE	4.84-1
4.28.8	MFAC	SMA2	4.28-3
4.24.8.13	MINMAX	PLØT	4.24-11
4.41.11.24	MPYL	SSG1	4.41.23
4.41.11.25	MPYLT	SSG1	4.41-23
4.28.8.21	MQDPLT	SMA2	4.28-8a
4.28.8.5	MRØD	SMA2	4.28-5
4.48.8.11	MTIMSU	READ	4.48-10
4.28.8.15	MTØRDR	SMA2	4.28-7
4.28.8.14	MTRAPR	SMA2	4.28-7
4.28.8.19	MTRBSC	SMA2	4.28-8
4.28.8.22	MTRIQD	SMA2	4.28-8a
4.28.8.13	MTRIRG	SMA2	4.28-7
4.28.8.20	MTRPLT	SMA2	4.28-8a
4.57.1	MTRXIN	MTRXIN	4.57-1
4.28.8.6	MTUBE	SMA2	4.28-5
4.41.11.20	NØRM	SSG1	4.41-21
4.48.8.10	NØRM1	READ	4.48-10
4.70.1	ØFP	ØFP	4.70-1
4.70.8.1	ØFPPUN	ØFP	4.70-2
4.70.8.2	ØFP1	ØFP	4.70-3
4.70.8.3	ØFP1A	ØFP	4.70-3
4.70.8.4	ØFP1BD	ØFP	4.70-3
4.70.8.5	ØFP2BD	ØFP	4.70-3
4.70.8.6	ØFP3BD	ØFP	4.70-3
4.70.8.7	ØFP4BD	ØFP	4.70-3
4.70.8.8	ØFP5BD	ØFP	4.70-3
4.48.8.5	ØRTCK	READ	4.48-7
4.59.8.13	ØRTHØ	CEAD	4.59-10
4.7.5.10	ØSCDMP	XGPI	4.7-6
4.24.8.1	PARAM	PLØT	4.24-4
4.83.1	PARTNX	PARTN	4.83-1
4.41.11.17	PERMUT	SSG1	4.41-20
4.24.8.14	PERPEC	PLØT	4.24-11
4.55.8.5	PKBAR	PLA4	4.55-4
4.55.8.10	PKQAD1	PLA4	4.55-5
4.55.8.11	PKQAD2	PLA4	4.55-5
4.55.8.7	PKQDM	PLA4	4.55-5
4.55.8.18	PKQDMS	PLA4	4.55-7
4.55.8.13	PKQDM1	PLA4	4.55-6
4.55.8.22	PKQDPL	PLA4	4.55-8
4.55.8.4	PKRØD	PLA4	4.55-4

Section Number	Entry Point	Module Name	Page Number
4.55.8.14	PKTQ1	PLA4	4.55-6
4.55.8.16	PKTQ2	PLA4	4.55-7
4.55.8.20	PKTRBS	PLA4	4.55-8
4.55.8.8	PKTRI1	PLA4	4.55-5
4.55.8.9	PKTRI2	PLA4	4.55-5
4.55.8.6	PKTRM	PLA4	4.55-4
4.55.8.17	PKTRMS	PLA4	4.55-7
4.55.8.12	PKTRM1	PLA4	4.55-6
4.55.8.21	PKTRPL	PLA4	4.55-8
4.55.8.19	PKTRQD	PLA4	4.55-7
4.55.8.15	PKTRQ2	PLA4	4.55-6
4.52.1	PLA1	PLA1	4.52-1
4.53.1	PLA2	PLA2	4.53-1
4.54.1	PLA3	PLA3	4.54-1
4.54.8.1	PLA31	PLA3	4.54-3
4.54.8.2	PLA32	PLA3	4.54-3
4.55.1	PLA4	PLA4	4.55-1
4.55.8.3	PLA4B	PLA4	4.55-4
4.55.8.1	PLA41	PLA4	4.55-3
4.55.8.2	PLA42	PLA4	4.55-3
4.41.11.13	PLØAD	SSG1	4.41-9
4.24.8.3	PLØT	PLØT	4.24-5
4.24.8.5	PLTØPR	PLØT	4.24-6
4.41.11.15	PRESAX	SSG1	4.41-20
4.24.8.15	PRØCES	PLØT	4.24-12
4.76.1	PRTMSG	PRTMSG	4.76-1
4.77.1	PRTPRM	PRTPRM	4.77-1
4.54.8.4	PSBAR	PLA3	4.54-4
4.54.8.9	PSQAD1	PLA3	4.54-5
4.54.8.10	PSQAD2	PLA3	4.54-5
4.54.8.6	PSQDM	PLA3	4.54-4
4.54.8.12	PSQDM1	PLA3	4.54-6
4.54.8.16	PSQPL1	PLA3	4.54-7
4.54.8.3	PSRØD	PLA3	4.54-3
4.54.8.15	PSTPL1	PLA3	4.54-6
4.54.8.13	PSTQ1	PLA3	4.54-6
4.54.8.18	PSTQ2	PLA3	4.54-7
4.54.8.14	PSTRB1	PLA3	4.54-6
4.54.8.7	PSTRI1	PLA3	4.54-5
4.54.8.8	PSTRI2	PLA3	4.54-5
4.54.8.17	PSTRQ2	PLA3	4.54-7
4.54.8.5	PSTRM	PLA3	4.54-4
4.54.8.11	PSTRM1	PLA3	4.54-5

Section Number	Entry Point	Module Name	Page Number
4.41.11.35	QDMEM	SSG1	4.41-27
4.19.1	QPARAM	PARAM	4.19-1
4.48.8.38	QRITER	READ	4.48-19d
4.64.1	RANDØM	RANDØM	4.64-1
4.64.8.4	RAND1	RANDØM	4.64-6
4.64.8.5	RAND2	RANDØM	4.64-6
4.64.8.6	RAND3	RANDØM	4.64-7
4.64.8.7	RAND4	RANDØM	4.64-7
4.64.8.2	RAND5	RANDØM	4.64-6
4.64.8.8	RAND6	RANDØM	4.64-7
4.64.8.1	RAND7	RANDØM	4.64-5
4.64.8.3	RAND8	RANDØM	4.64-4
4.37.1	RBMG1	RBMG1	4.37-1
4.38.1	RBMG2	RBMG2	4.38-1
4.39.1	RBMG3	RBMG3	4.39-1
4.40.1	RBMG4	RBMG4	4.40-1
4.48.8.1	READ1	READ	4.48-4
4.48.8.2	READ2	READ	4.48-5
4.48.8.3	READ3	READ	4.48-6
4.48.8.4	READ4	READ	4.48-7
4.48.8.42	READ6	READ	4.48-19f
4.48.1	REIG	READ	4.48-1
4.41.11.14	RFØRCE	SSG1	4.41-19
4.27.8.34	RØMBDK	SMA1	4.27-19
4.46.8	RØMBER	SDR2	4.46-7
4.48.8.35	RØTATE	READ	4.48-19c
4.48.8.34	RØTAX	READ	4.48-19c
4.4.5.2	RPAGE	XSØRT	4.4-3
4.46.8.15	SBAR1	SDR2	4.46-12
4.46.8.32	SBAR2	SDR2	4.46-18
4.46.8.5	SBEAM1	SDR2	4.46-10
4.46.8.24	SBEAM2	SDR2	4.46-16
4.46.8.27	SBSPL2	SDR2	4.46-16
4.35.1	SCE1	SCE1	4.35-1
4.46.8.16	SCØNE1	SDR2	4.46-12
4.46.8.30	SCØNE2	SDR2	4.46-17
4.46.8.31	SCØNE3	SDR2	4.46-18
4.45.1	SDR1	SDR1	4.45-1
4.45.8.1	SDR1A	SDR1	4.45-5
4.45.8.2	SDR1C	SDR1	4.45-5
4.45.8.3	SDR1D	SDR1	4.45-5
4.46.1	SDR2	SDR2	4.46-1

Section Number	Entry Point	Module Name	Page Number
4.46.8.2	SDR2A	SDR2	4.46-8
4.46.8.1	SDR2AA	SDR2	4.46-8
4.46.8.3	SDR2B	SDR2	4.46-9
4.46.8.20	SDR2C	SDR2	4.46-13
4.46.8.21	SDR2D	SDR2	4.46-14
4.46.8.22	SDR2E	SDR2	4.46-15
4.62.1	SDR3	SDR3	4.62-1
4.62.8.1	SDR3A	SDR3	4.62-9
4.74.1	SEEMAT	SEEMAT	4.74-1
4.46.8.13	SELAS1	SDR2	4.46-12
4.46.8.26	SELAS2	SDR2	4.46-16
4.23.8.1	SETINP	PLTSET	4.23-3
4.20.1	SETVAL	SETVAL	4.20-1
4.24.8.8	SHAPE	PLØT	4.24-8
4.48.8.32	SICØX	READ	4.48-19b
4.48.8.33	SINCAS	READ	4.48-19b
4.41.11.12	SLØAD	SSG1	4.41-19
4.27.1	SMA1	SMA1	4.27-1
4.27.8.2	SMA1A	SMA1	4.27-8
4.27.8.3	SMA1B	SMA1	4.27-9
4.27.8.4	SMA1BD	SMA1	4.27-9
4.28.1	SMA2	SMA2	4.28-1
4.28.8.2	SMA2A	SMA2	4.28-4
4.28.8.3	SMA2B	SMA2	4.28-4
4.28.8.4	SMA2BD	SMA2	4.28-4
4.30.1	SMA3	SMA3	4.30-1
4.30.8.1	SMA3A	SMA3	4.30-5
4.30.8.2	SMA3B	SMA3	4.30-5
4.30.8.4	SMA3BD	SMA3	4.30-6
4.30.8.3	SMA3C	SMA3	4.30-6
4.48.8.30	SMLEIG	READ	4.48-19a
4.36.1	SMP1	SMP1	4.36-1
4.50.1	SMP2	SMP2	4.50-1
4.80.1	SØLVE	SØLVE	4.80-1
4.46.8.7	SPANL1	SDR2	4.46-10
4.46.8.25	SPANL2	SDR2	4.46-16
4.46.8.12	SQDME1	SDR2	4.46-11
4.46.8.10	SQDPL1	SDR2	4.46-11
4.46.8.27	SQRTM	READ	4.48-19
4.46.8.4	SRØD1	SDR2	4.46-10
4.46.8.23	SRØD2	SDR2	4.46-16

Section Number	Entry Point	Module Name	Page Number
4.41.1	SSG1	SSG1	4.41-1
4.41.11.1	SSG1A	SSG1	4.41-14
4.42.1	SSG2	SSG2	4.42-1
4.42.8.1	SSG2B1	SSG2	4.42-4
4.43.1	SSG3	SSG3	4.43-1
4.44.1	SSG4	SSG4	4.44-1
4.65.8.7	STEP	TRD	4.65-15
4.46.8.19	STØRD1	SDR2	4.46-13
4.46.8.35	STØRD2	SDR2	4.46-19
4.46.8.28	STQME2	SDR2	4.46-17
4.46.8.18	STRAP1	SDR2	4.46-13
4.46.8.34	STRAP2	SDR2	4.46-18
4.46.8.8	STRBS1	SDR2	4.46-10
4.46.8.17	STRIR1	SDR2	4.46-13
4.46.8.33	STRIR2	SDR2	4.46-18
4.46.8.11	STRME1	SDR2	4.46-11
4.46.8.9	STRPL1	SDR2	4.46-11
4.46.8.14	STRQD1	SDR2	4.46-12
4.46.8.29	STRQD2	SDR2	4.46-17
4.46.8.6	STUBE1	SDR2	4.46-10
4.48.8.13	SUB	READ	4.48-11
4.48.8.26	SUMM	READ	4.48-18
4.3.7.7	SWSRT	IFP1	4.3-6
4.75.1	TABPT	TABPT	4.75-1
4.26.1	TA1	TA1	4.26-1
4.26.8.2	TA1A	TA1	4.26-14
4.26.8.3	TA1B	TA1	4.26-14
4.26.8.4	TA1C	TA1	4.26-14
4.26.8.5	TA1CA	TA1	4.26-15
4.26.8.6	TA1F	TA1	4.26-15
4.26.8.7	TA1G	TA1	4.26-15
4.41.11.3	TEMPL	SSG1	4.41-15
4.41.11.10	TPØNT	SSG1	4.41-18
4.65.1	TRD	TRD	4.65-1
4.65.8.1	TRD1A	TRD	4.65-12
4.65.8.2	TRD1B	TRD	4.65-12
4.65.8.4	TRD1C	TRD	4.65-13
4.65.8.9	TRD1D	TRD	4.65-16
4.65.8.10	TRD1E	TRD	4.65-16
4.48.8.31	TRIDI	READ	4.48-19a
4.41.11.36	TRIMEM	SSG1	4.41-27

Section Number	Entry Point	Module Name	Page Number
4.85.1	TRNSP	TRNSP	4.85-1
4.41.11.32	TTØRDR	SSG1	4.41-26
4.41.11.31	TTRAPR	SSG1	4.41-25
4.41.11.30	TTRIRG	SSG1	4.41-25
4.8.1	UMFEDT	UMFEDIT	4.8-1
4.8.6	UMFZBD	UMFEDIT	4.8-1
4.48.8.29	VALVEC	READ	4.48-19
4.60.8.1	VDR	VDR	4.60-5
4.60.8.2	VDRA	VDR	4.60-5
4.60.8.3	VDRB	VDR	4.60-6
4.60.9.2	VDRBD	VDR	4.60-7
4.48.8.39	WILVEC	READ	4.48-19e
4.76.8.2	WRTMSG	PRTMSG	4.76-2
4.4.5.5	XBCDBI	XSØRT	4.4-4
4.11.1	XCEI	REPT	4.11-1
4.12.1	XCEI	JUMP	4.12-1
4.13.1	XCEI	CØND	4.13-1
4.14.1	XCEI	EXIT	4.14-1
4.18.1	XCEI	END	4.18-1
4.10.1	XCHK	XCHK	4.10-1
4.9.5.2	XCLEAN	XSFA	4.9-4
4.2.1	XCSA	XCSA	4.2-1
4.9.5.4	XDPH	XSFA	4.9-6
4.17.1	XEQUIV	EQUIV	4.17-1
4.4.5.4	XFADJ	XSØRT	4.4-4
4.4.5.10	XFADJ1	XSØRT	4.4-5
4.9.5.7	XFILPS	XSFA	4.9-7
4.7.5.9	XFLDEF	XGPI	4.7-5
4.7.5.8	XFLØRD	XGPI	4.7-5
4.7.1	XGPI	XGPI	4.7-1
4.7.6.2	XGPIBD	XGPI	4.7-7
4.7.5.2	XGPIBS	XGPI	4.7-3
4.7.5.1	XGPIDG	XGPI	4.7-3
4.7.5.1	XGPIMW	XGPI	4.7-3
4.7.5.5	XIPFL	XGPI	4.7-4
4.7.5.4	XLNKHD	XGPI	4.7-4
4.7.5.5	XØPFL	XGPI	4.7-4
4.7.5.3	XØSGEN	XGPI	4.7-3
4.7.5.6	XPARAM	XGPI	4.7-4
4.9.5.6	XPLEQK	XSFA	4.9-6
4.9.5.5	XPØLCK	XSFA	4.9-6

Section Number	Entry Point	Module Name	Page Number
4.4.5.6	XPRETY	XSØRT	4.4-4
4.9.5.3	XPUNP	XSFA	4.9-5
4.16.1	XPURGE	XPURGE	4.16-1
4.4.5.1	XRECPS	XSØRT	4.4-3
4.2.5.1	XRGDFM	XCSA	4.2-1
4.15.1	XSAVE	XSAVE	4.15-1
4.2.5.4	XSBSET	XCSA	4.2-2
4.7.5.7	XSCNDM	XGPI	4.7-4
4.9.1	XSFA	XSFA	4.9-1
4.4.1	XSØRT	XSØRT	4.4-1
4.9.5.1	XSØSGN	XSFA	4.9-3
4.48.8.12	XTRNSY	READ	4.48-11
4.63.8.1	XYDUMP	XYTRAN	4.63-5
4.63.8.2	XYFIND	XYTRAN	4.63-5
4.63.8.4	XYLØG	XYTRAN	4.63-6
4.63.8.3	XYØUT	XYTRAN	4.63-6
4.69.1	XYPLØT	XYPLØT	4.69-1
4.63.8.5	XYTICS	XYTRAN	4.63-7
4.63.1	XYTRAN	XYTRAN	4.63-1

4.2 EXECUTIVE PREFACE MODULE XCSA (EXECUTIVE CONTROL SECTION ANALYSIS)

4.2.1 Entry Point: XCSA

4.2.2 Purpose

To process the NASTRAN Executive Control Deck.

4.2.3 Calling Sequence

CALL XCSA. XCSA is called only by subroutine SEMINT.

4.2.4 Method

The cards of the Executive Control Deck are read and processed with checks being made for illegal formats, duplication and errors peculiar to the particular card being processed. When all of the control cards have been processed (i.e., CEND control card found), the Executive Control Table (XCSA) is written on the Problem Tape and XCSA returns to the calling routine.

4.2.5 Subroutines

4.2.5.1 Subroutine Name: XRGDFM

1. Entry Point: XRGDFM
2. Purpose: To select a rigid format based on the SØL card in the Executive Control Deck.
3. Calling Sequence: CALL XRGDFM (NEWSØL,ØLDSØL,IAPP)

   NEWSØL - Two-word array containing solution and subset numbers taken from SØL control card.

   ØLDSØL - Two-word array containing solution and subset numbers taken from the Old Problem Tape if the problem is a restart. If not a restart, ØLDSØL = 0.

   IAPP - Approach code (1 = FØRCE, 2 = DISPL, 3 = DMAP) taken from the APP card in the Executive Control Deck.

4. Method: If the problem is being restarted, a check is made for a solution (Rigid Format) change. If the solution has been changed, a bit is set in table MEDMSK in named common block /XMDMSK/.

A check is made for a legal solution number, and, if acceptable, a branch is made on the solution number, and subroutine LDi (i = solution number) is called to create the DMAP and MED records for the XCSA table. XRGDFM then returns to the calling routine XCSA.

4.2.5.2 Subroutine Name: LDi, where i = solution number, i = 01, 02, ..., 12.

1. Entry Point: LDi

2. Purpose: To write the DMAP sequence and MED records of the XCSA Executive Table for solution (Rigid Format) i (see XCSA Executive Control Table description, section 2.4.2.5).

3. Calling Sequence: CALL LDi (SUBSET)

SUBSET - Solution subset number from the SØL control card.

4. Method: The packed DMAP program is generated, and then subroutine XSBSET is called to select the proper solution subset for the DMAP program by altering the IS1 array. Upon return from XSBSET the arrays IS1, JNM and INM (see 4.2.6.2 below) are written on the New Problem Tape to complete the MED record for Executive Table XCSA. LDi then returns to calling routine.

5. Restrictions: Due to restrictions imposed by various FØRTRAN compilers, lengthy Rigid Formats (e.g., Piecewise Linear Analysis (i = 06), Direct Complex Eigenvalue Analysis (i = 08) have required auxiliary subroutines (e.g., L061 auxiliary to LD06 and L081 auxiliary to LD08, see section 4.2.5.3) to contain the Rigid Format.

4.2.5.3 Block Data Subprograms LiBD and LiABD

Associated with each LDi subroutine are the block data programs LiBD and LiABD for i = 01, 02, ..., 12. While the LDi subroutine contains the i^{th} Rigid Format in FØRTRAN DATA statements, the LiBD block data subprogram contains bit numbers for the Card Name Restart Table and bit numbers for the File Name Restart Table for the i^{th} Rigid Format, and the LiABD block data subprogram contains the general Restart Tables for the i^{th} Rigid Format.

4.2.5.4 Subroutine Name: XSBSET

1. Entry Point: XSBSET

2. Purpose: To eliminate DMAP instructions not belonging to the specified subset by altering the IS1 array.

3. Calling Sequence: CALL XSBSET (ID1,NSS,SUBSET,IS1,NDI,NWPI)

   ID1 - Table containing DMAP instruction numbers of those instructions that are not part of the specified subset.

   NSS - Number of subsets in table ID1.

   SUBSET - Subset to be selected from table ID1.

   IS1 - Module execution decision table.

   NDI - Number of DMAP instructions in DMAP program.

   NWPI - Number of words per IS1 entry.

4. Method: Table ID1 is searched and the proper subset is selected. Each DMAP instruction has a corresponding entry in table IS1. If the IS1 entry for an instruction is zero, then the instruction is eliminated from the DMAP program. Therefore zeroing the IS1 entries of those instructions specified in table ID1 yields the proper subset. XSBSET then returns to the calling routine.

4.2.6 Design Requirements

4.2.6.1 Use of Open Core

Open core is used for GINØ buffers, for generating the XPTDIC Executive Table (see section 2.4) on restarts, and for storing user generated DMAP programs. Named common block /XCSABF/ defines the beginning of open core for module XCSA. Since XPTDIC is not stored permanently in open core and because the use of open core to store a DMAP program and a call to LDi are mutually exclusive, the LDi subroutines can be origined for overlay purposes at the same location as /XCSABF/.

4.2.6.2 Restart Tables Initialized in the Routines

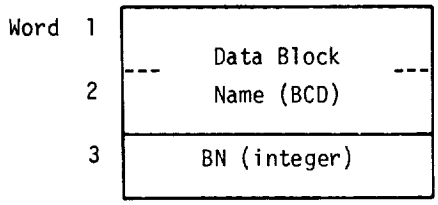
The following tables are initialized by the LDi (i = solution number) subroutine or its associated Block Data program and are used to aid module XGPI in restarting a problem which uses Rigid Format i.

1. IS1 - Module Execution Decision Table: This table when used in conjunction with table MEDMSK in named common block /XMDMSK/ will provide module XGPI with the information needed to decide whether or not to set the execute flag in an ØSCAR entry. Each DMAP instruction

in a Rigid Format has a corresponding entry in ISl. An entry in ISl can be one to five words in length, and only bits 1 through 31 are used to form a truth table. Note that if an ISl entry is zero, the corresponding DMAP instruction is unconditionally excluded from the Rigid Format DMAP program being compiled by module XGPI.

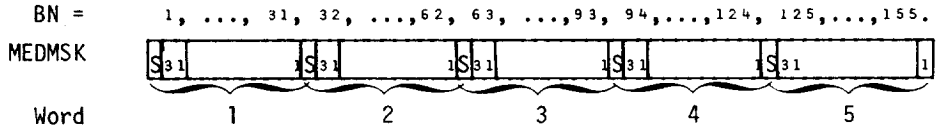
2. JNM - File Name Restart Table: The JNM table provides module XGPI with the capability of regenerating data blocks which are missing in the restart dictionary and which are needed to restart the problem. Note that the restart must be a modified restart. If a data block is missing, the JNM table will indicate which bit to set in table MEDMSK. MEDMSK is then used with ISl to determine which DMAP modules must be re-executed in order to correctly regenerate the missing data block.

Sample JNM Entry:



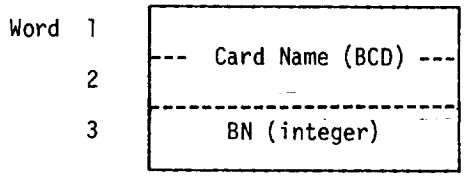
BN is the bit number of the bit which is to be set in table MEDMSK to regenerate the specified data block. The usable bits (bits 1-31) of MEDMSK are numbered sequentially starting from bit 31 of the first word. MEDMSK is five words long.

Example:



3. INM - Card Name Restart Table: When the problem is being restarted and input data (Bulk Data and/or Case Control Data) has been modified, table INM tells module XGPI whether or not the modifications affect the compilation of the DMAP program associated with the Rigid Format. Table MJCD in named common block /IFPX1/ and table MJMSK in named common block /IFPXO/ indicate which cards have been modified. If INM has an entry for a modified card, the INM entry will indicate which bit to set in table MEDMSK. MEDMSK is then used with ISl to determine which DMAP modules must be re-executed.

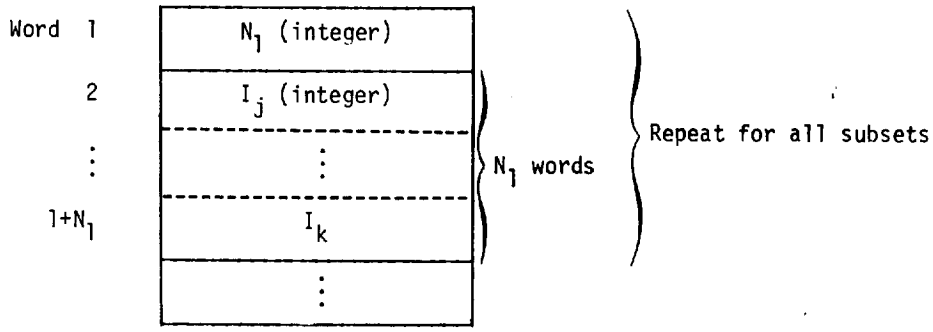
Sample INM Entry:



BN is the number of the bit which is to be set in table MEDMSK if the associated card name has been modified. See sample JNM entry for further description of BN.

4. ID1 - Subset Table: DMAP instructions in a DMAP program are numbered sequentially starting with "BEGIN" as instruction number 1. Table ID1 contains the instruction numbers of those instructions that are <u>not</u> to be included in a subset.

Sample ID1 Entry:



N_1 = number of instructions to delete in subset 1. ($N_1 \geq 0$).

I_j - I_k = DMAP instruction numbers of instructions to be deleted.

4.2.7 Diagnostic Messages

Every effort is made to get through module XCSA so that the modules IFP1 and IFP can process the Case Control Deck and the Bulk Data Deck. XCSA sets the NØGØ flag in named common block /SYSTEM/ according to the severity of the errors found.

NØGØ = 0 - no errors found.

1 - job will terminate after module XGPI.

2 - job will terminate after IFP modules.

3 - job terminates in XCSA.

See diagnostic message section of User's Manual (section 6.2) for a detailed discussion of XCSA error messages. XCSA messages include numbers 501 thru 526.

4.3 EXECUTIVE PREFACE MODULE IFP1 (INPUT FILE PROCESSOR, PART 1)

4.3.1 Entry Point: IFP1

4.3.2 Purpose

To process the Case Control Deck. See section 2.3 of the User's Manual for a discussion of the Case Control Deck.

4.3.3 Calling Sequence

CALL IFP1. IFP1, a Preface module, is called only by subroutine SEMINT.

4.3.4 Input Data

The input data consists of the Case Control Deck and the CASECC data block from the Old Problem Tape if the problem is a restart.

4.3.5 Output Data Blocks

CASECC - Case Control Data Table.

PCDB - Plot Control Data Table (for the structure plotter).

XYCDB - XY output Control Data Block.

Notes:

1. CASECC will always exist.
2. PCDB will exist only if a structure plotter package is included in the Case Control Deck.
3. XYCDB will exist only if a XYØUT plotter package is included in the Case Control Deck.

4.3.6 Method

The Case Control Cards are read and stored on a scratch file for later use. The title cards are abstracted to form page headings. Title card abstraction is stopped by a SYM, SUBCASE, SYMCØM or REPCASE card. IFP1 is stopped by a BEGIN BULK card.

The construction of CASECC is as follows:

1. The scratch tape is read one card at a time, and subroutine XRCARD is called to translate the card.

2. The first four characters beginning with a non-blank are identified in a key word table, and card type dependent routines are executed. (See Case Control Deck in User's Manual).

3. When "SUBCASE" type cards are encountered, a CASECC record is written out.

4. If the card ØUTPUT (PLØT) is encountered,XRCARD images of succeeding cards are written on PCDB.

5. If the card ØUTPUT (XYØUT) is encountered,IFP1XY processes the succeeding cards into the XYCDB.

The module conclusion is as follows:

1. A copy of CASECC is placed on the NPTP for use in restart.

2. If this run is a restart, IFP1B is called to analyze CASECC changes and set modify flags for later use in Executive Preface module XGPI (see section 4.7).

4.3.7 Subroutines

4.3.7.1 Subroutine Name: IFP1B

1. Entry Point: IFP1B

2. Purpose: To set modify flags for use in modified restart.

3. Calling Sequence: CALL IFP1B (ICASE,ØPTP,CASECC,IBUF1,IBUF2,LENCC)

   ICASE - A two-dimensional array (LENCC,2) for storage of both copies of CASECC - array - input.

   ØPTP - GINØ file name of the Old Problem Tape - BCD - input.

   CASECC - GINØ file name of CASECC - BCD - input.

   IBUF1, IBUF2 - GINØ buffer pointers - integer - input.

   LENCC - Row dimension of ICASE - integer - input.

COMMON/IFPXO/SPACE(3),NWORDS

NWORDS - Pointer into /IFPXO/ such that IFP1 modify flags are in SPACE(NWORDS).

4. Method:

CASECC and the copy of CASECC on the OPTP are compared according to the following scheme. The local array IWORD classifies each word in CASECC into 0 and 1. 0 words: If the CASECC word is non-zero and IBIT is non-zero in this position, the IBIT bit is turned on in /IFPXO/. 1 word: If the CASECC word is different from the OPTP word and IBIT is non-zero in this position, the IBIT is turned on in /IFPXO/.

IFP1B also determines the loop nature of the problem. The looping rules are as follows:

a. The current problem will loop under the following conditions: SPC set changes; MPC set changes; direct input matrix changes; transfer function set changes; transient load changes; differential stiffness coefficient set is greater than zero; and Piecewise Linear coefficient set is greater than zero. If any of the above conditions are met, LOOP$ is turned on in /IFPXO/.

b. The old problem might have been a looping problem if the above conditions were present in the OPTP CASECC. If the old problem was a looping problem as determined in (a) and the number of records in CASECC changes, LOOP1$ is set (this should force the re-execution of the entire loop).

c. If the problem is not a looping problem, NOLOOP$ is set.

4.3.7.2 Subroutine Name: IFP1C

1. Entry Point: IFP1C

2. Purpose: To construct set lists from SET cards.

3. Calling Sequence: CALL IFP1C (ISUB,I81,CORE,SCR1,NWPC,ICC,NZ,THRU,NSET)

ISUB - 1 - master set. 2 - set belongs to a subcase - integer - input.

I81 - Pointer to storage for set in core - integer - input/output.

CORE - Open core array.

SCR1 - GINO file number of scratch file containing card images - integer - input.

NWPC - Number of words per card - integer - input.

ICC - Current line count of Case Control card echo - integer - input/output.

NZ - Length of open core - integer - input/output.

THRU - BCD value "THRU" - BCD - input.

NSET - Number of sets found to current set - integer - input.

4.3.7.3 Subroutine Name: IFP1D

1. Entry Point: IFP1D

2. Purpose: To write user diagnostic messages from IFP1.

3. Calling Sequence: CALL IFP1D (MSGN)

MSGN - User message number - integer - input.

4.3.7.4 Subroutine Name: IFP1E

1. Entry Point: IFP1E

2. Purpose: To write out CASECC and update sets.

3. Calling Sequence: CALL IFP1E (CASE,ISUBC,SYMSEQ,NWDSC,I81)

CASE - Array containing Case Control record to be written out (CASE (LENCC,2)).

ISUBC - Five word BCD array containing current subcase number - BCD - output.

SYMSEQ - Symmetry coefficient array - real - input.

NWDSC - Pointer to beginning of set lists - integer - input/output.

I81 - Pointer to end of set list - integer - output.

4.3.7.5 Subroutine Name: IFP1F

1. Entry Point: IFP1F

2. Purpose: To find the first four characters beginning with a non-blank on one input card.

3. Calling Sequence: CALL IFP1F ($n,IWØRD,IS,IBEN,II)

$n - FØRTRAN statement number which defines the return to be taken if the entire card is blank.

IWØRD - First four characters beginning with a non-blank, left justified - BCD - output.

IS - Number of bits/character times (number of characters/word-1) - integer - input.

IBEN - Mask used to determine if character is blank 'b000' - input.

II - Pointer to word in which IWØRD begins - integer - output.

CØMMØN/IFP1X/CØRE(20)

IFP1X - 20-word array holding card image - BCD - input.

CØMMØM/IFP1A/

IFP1A - See /IFP1A/ description under Design Requirements (section 4.3.8).

4.3.7.6 Subroutine Name: IFP1G

1. Entry Point: IFP1G

2. Purpose: To find an equal sign and copy the remainder of the data into a specified arrays

3. Calling Sequence: CALL IFP1G (ITYPE,CASE,ISUB1)

ITYPE - Indicates area in which to store data.

1. TITLE
2. SUBTITLE
3. LABEL
4. HEAD1
5. HEAD2
6. HEAD3
7. PLØTID

} of /ØUTPUT/

8. First 32 words of CASE

- integer - input.

CASE - Case control array (132,2) unless ITYPE = 8, when it may be only 32 word array.

ISUB1 - Subcase number -1 or 2 of CASE array.

CØMMØN/IFP1X/CØRE(20)

IFP1X - 20-word array holding card image.

COMMON/OUTPUT/

OUTPUT - Output common block - holds BCD titles for NASTRAN pages.

COMMON/IFP1A/

IFP1A - See /IFP1A/ description under Design Requirements (section 4.3.8).

4.3.7.7 Subroutine Name: SWSRT

1. Entry Point: SWSRT

2. Purpose: To check set lists for duplicates and overlapping intervals. SWSRT also sorts lists into increasing order.

3. Calling Sequence: CALL SWSRT (LIST,ISTOR,NLIST)

LIST - Array of set members.

ISTOR - Scratch space of length NLIST.

NLIST - Number of members in LIST.

4.3.7.8 Subroutine Name: IFP1XY

1. Entry Point: IFP1XY

2. Purpose: To construct the XYCDB data block.

3. Calling Sequence: CALL IFP1XY (FLAG)

FLAG - 0, first entry for initialization; 1, data entry; -1, last entry. FLAG is set to 1 on return from the first entry - integer - input/output.

COMMON/IFP1X/CARD(20), CARD1(20)

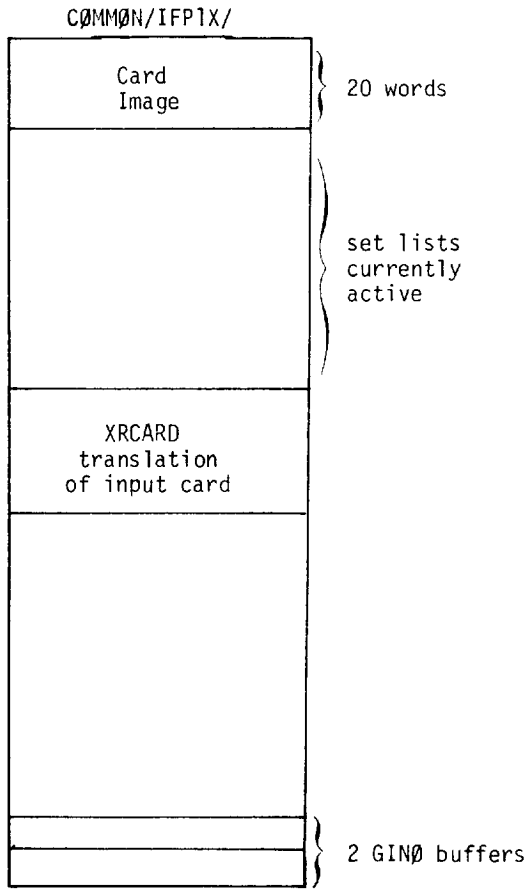
CARD - Contains card data as read from the input file.

CARD1 - Contains XRCARD translation of CARD unless CARD(1) contains 'XTIT', 'YTIT', 'TCUR', 'YTTI' or 'YBTI'. In this case CARD1 (20) contains BCD data occurring after the equal sign.

4.3.8 Design Requirements

1. One scratch file.

2. Open core at /IFP1X/.



3. Common block IFP1A.

Name	Length	Meaning	Initialized to
NAME	2	Name of data block for error messages.	CASE CC
SUBC	1	Subcase key word	SUBC
SET	1	Set key word	SETb
SYMS	1	Symmetry sequence key word	SYMS
TSTE	1	Time step card selection key word	TSTE
LABE	1	Label key word	LABE
SUBT	1	Subtitle key word	SUBT
SCR1	1	GINØ file number of scratch file	301
CASECC	1	GINØ file name of CASECC	CASE
BLANK	1	Blank word	bbbb

Name	Length	Meaning	Initialized to
CARD	1	Heading word card	CARD
CØUN	1	Heading word CØUNT	CØUN
T	1		T
BEGI	1	Begin bulk key word	BEGI
TITL	1	Title key word	TITL
CASEN	11	Case Control page heading	Case Control Deck echo
SPCF	1	Forces of constraint key word	SPCF
VELØ	1	Velocity key word	VELØ
ACCE	1	Acceleration key word	ACCE
ELFØ	1	Element forces key word	ELFØ
STRE	1	Element stress key word	STRE
DISP	1	Displacement key word	DISP
ØUTP	1	Øutput key word	ØUTP
SYM	1	Symmetry subcase key word	SYM
FREQ	1	Frequency set key word	FREQ
DLØA	1	Dynamic loading key word	DLØA
TEMP	1	Temperature field key word	TEMP
DEFØ	1	Deformation set key word	DEFØ
TIME	1	Time key word	TIME
SPC	1	Single-point constraint set key word	SPC
MAXL	1	Maximum number of output lines key word	MAXL
IC	1	Initial condition set selection id	IC
METH	1	Real eigenvalue or buckling method selection key word	METH
LØAD	1	Load set selection key word	LØAD
MPC	1	Multipoint set selection key word	MPCb
STIF	1	Stiffness thermal field key word	STIF
ALL	1	ALL key word	ALLb
THRU	1	THRU key word	THRU
SØRT	1	Sorted echo key word	SØRT
UNSØ	1	Unsorted echo key word	UNSØ

Name	Length	Meaning	Initiated to
ECHØ	1	Bulk data echo key word	ECHØ
PLØT	1	Plot key word	PLØT
MØDE	1	Modes key word	MØDE
PUNC	1	Punch key word	PUNC
PRIN	1	Print key word	PRIN
NWPC	1	Number of words per card	20
NCPW	1	Number of characters per word (NASTRAN)	4
BØTH	1	Echo-sorted and unsorted	BØTH
NØNE	1	None key word	NØNE
PCDB	1	Plot control data GINØ file name	PCDB
NAME	2	GINØ error message for PCDB file	PCDB bbbb
VECT	1	Alternate displacement key word	VECT
SYMC	1	Symcom key word	SYMC
EQUAL1	1	Equal sign left adjusted	=bbb
NMØDES	1	Value of modes card	1
IBØB	1	Structure plot flag 0, not currently in structure plot mode 1, in structure plot mode	0
IEND	1		0
ISYMCM	1	Symcom flag	0
LØADN	1	Current Subcase ID number	1
IØUT2	1		0
ICC	1	Printed card count	1
NSET	1	Number of current set lists	0
NSYM	1	Number of 'SYM' subcases	1
INØMØR	1	Flag to turn off Title card search	0
NPTP	1	GINØ file name of the New Problem Tape	NPTP
ØPTP	1	GINØ file name of the Old Problem Tape	ØPTP
DØL	1	Dollar sign	$bbbb
ZZZZBB	1	Hollerith zeros	0000bb

Name	Length	Meaning	Initiated to
ISTR	1	Storage flag for IFP1G titles	1
ISUB	1	Subcase or master CASECC pointer	1
K2PP	1	Key words for direct input matrix selection	K2PP
B2PP	1		B2PP
M2PP	1		M2PP
DSCØ	1	Key word for differential stiffness set selection	DSCØ
REPC	1	Key word for repeat subcase subcase	REPC
LENCC	1	Length of Case Control Record	166
LINE	1	Key word for LINE/page count	LINE
ØM	1	Word to distinguish between SUBCØM SUBCASE	ØMbb
TFL	1	Key word for transfer function set selection	TFL
DEFA	1	Key word for default specification	DEFA
ELST	1	Key word for element stress set selection	ELST
MAT	1	Key word for thermal material set selection	MATE
ØFRE	1	Key word for output frequency set selection	ØFRE
IMAG	1	Key word for real/imaginary printout	IMAG
PHAS	1	Key word for magnitude/phase printout	PHAS
REAL	1	Key word for real or real/imaginary printout	REAL
CMET	1	Key word for complex eigenvalue set selection	CMET
SDAM	1	Key word for Structural Damping Table for use in modal formulation	SDAM
INER	1	Key word for Inertia Relief Element set selection	INER
ADIS	1	Key word for solution set displacement selection	SDIS
AVEL	1	Key word for solution set velocity selection	SVEL
AACC	1	Key word for solution set acceleration selection	SACC
NØNL	1	Key word for non-linear load set selection	NØNL
CØNF	1	Not used	
XYPL	1	Key word for XYPLØT packet delimiter	XYPL
PLCØ	1	Key word for Piecewise Linear set selection	PLCØ

Name	Length	Meaning	Initiated to
AXIS	1	Key word for selection of Axis symmetric boundary condition	AXIS
NLLØ	1	Key word for non-linear output set selection	NLLØ
DELE	1	Key word for element deletion set selection	DELE
XYCB	1	GINØ file name of XY control data block	XYCB
ØNEB	1	BCD one	1bbb
HARM	1	Key word for harmonic output control	HARM
SINE	1	Key word for sine boundary conditions	SINE
COSI	1	Key word for cosine boundary conditions	COSI
SUBS	1	Key word for SUBSEQ	SUBS
AVEC	1	Key word for solution set vector output	SVEC
FØRC	1	Not used	
RAND	1	Key word for random set selection	RAND
XYØU	1	Key word for XYPLØT packet delimiter	XYØU
ØLØA	1	Key word for output load set selection	ØLØA
PLT1	1	GINØ file name of BCD plot tape	PLT1
PLT2	1	GINØ file name of binary plot tape	PLT2
XTIT	1	Key words for XY output titles	XTIT
YTIT	1		YTIT
TCUR	1		TCUR
YTTI	1		YTTI
YBTI	1		YBTI
IBEN		Right shifted blank '000b'	--
EQUAL		Right shifted equal '000='	--

4. Interface with /SYSTEM/ (see section 2.4).

IFP1 can set the following cells of SYSTEM:

a. NØGØ - (NØGØ flag). If a fatal error is detected.

b. NLPP - (Number of lines per page). If a LINE card is supplied by the user.

c. STFTEM - (Material Temperature Set ID). If a TEMP(MATE) card is supplied.

d. ECHØ - (Echo flag). If an ECHØ request is made.

e. SPLØTS - (Structure Plot Request). If a structure plot packet is requested.

f. PLØTF - (Plot Flag). If a structure plot request is present.

5. Interface with /ØUTPUT/.

IFP1 supplies the problem title, subtitle, and label as well as the Plot ID.

4.3.9 Diagnostic Messages

IFP1 makes every attempt to process the entire Case Control Deck so that the complete Preface will run. Hence all fatal messages only cause the NØGØ flag to turn on.

IFP1 causes messages 601-699. For the exact nature of these messages, refer to the Diagnostic Message section of the User's Manual.

4.4 EXECUTIVE PREFACE MODULE XSØRT (EXECUTIVE BULK DATA CARD SØRT)

4.4.1 Entry Point: XSØRT

4.4.2 Purpose

The function of XSØRT is to prepare a file on the New Problem Tape containing the sorted bulk data. The operation of XSØRT is influenced by the type of run. If a cold start, the bulk data is read from the system input stream (or the User's Master File), sorted and written on the New Problem Tape. If an unmodified restart, the bulk data is copied from the Old Problem Tape onto the New Problem Tape. If a modified restart, the bulk data is read from the Old Problem Tape, and cards are deleted and/or added in accordance with cards in the system input stream. Additionally, flags are set within restart tables for each card type changed in any way. Again, the sorted bulk data is written onto the New Problem Tape. A print of the unsorted and/or sorted bulk data is made on request. XSØRT processes all data cards between the BEGIN BULK and ENDDATA cards in the input stream. Both cards must be present to properly bracket the NASTRAN Bulk Data Deck.

4.4.3 Calling Sequence

CALL XSØRT. XSØRT, a Preface module, is called only by the Preface driver, SEMINT.

4.4.4 Method

If the input is to be from a User Master File, XSØRT begins by positioning the file to the beginning of the proper subset of bulk data cards. INITCØ is then called to initialize machine dependent masks and constants. The open core below XSØRT (/ESØRT/) is divided into 5 GINØ buffers and a work buffer. This work buffer will contain each data card and a chaining pointer to indicate its sorted position. That is, the cards will be placed into the work buffer in the same order as read, but their sorted order will be shown by a chaining word with each card pointing to the position of the next card in alphanumeric sort. If the work buffer is unable to hold all of the bulk data cards, each subset that fills the buffer is unchained and written in sorted order onto a scratch file. This writing onto a scratch file frees the work buffer for another subset of data cards.

Three scratch files may become involved in sorting a large number of bulk data cards. After the first two scratches are filled with sorted subsets, they are merged, while maintaining the sorted order, onto a third scratch. From this point, after each new subset is written onto a scratch, it is merged with the scratch containing all previous subsets. As an example, assume three scratches are named A, B, and C. Scratch A is written with the first subset of data from a filled work buffer. Scratch B is written with the second subset. Scratch A and B are then merged to form scratch C. This frees scratch A and B. Scratch A is then written with the third subset of data. A and C are merged to form a new B. A is then written with the fourth subset. A and B are merged to form a new C. This process continues until all bulk data has been sorted. Following the final merge, one of the scratch files will contain all of the sorted bulk data cards.

As the sort and merge operations are being performed, any continuation cards or delete cards encountered are written onto separate holding files. After all data cards in the input stream have been processed, each of these holding files is processed. The delete card values are placed in ascending order and any overlaps or redundancies are removed. The continuation cards are checked for duplication and an in-core dictionary of their connection words is formed.

XSØRT may now make a pass through the scratch file containing all of the sorted bulk data cards within the input stream. During this pass, the User Master File (UMF) or the Old Problem Tape (ØPTP) data cards are merged with those from the input stream. Both the UMF and ØPTP data cards were properly sorted during their preparation. As this merge progresses, any data cards designated for removal by delete control cards are discarded. If the NASTRAN run for which XSØRT is operating is a restart, all data cards within the input stream plus any deleted from the ØPTP will cause data card type flags to be set within restart tables. This entire pass is not performed if the run does not require either a UMF or ØPTP.

Now a final pass of the resulting sorted data is made to introduce any continuation cards and write the completed Bulk Data Deck onto the New Problem Tape. The continuation cards are connected to the sorted data cards by matching connection words. Continuation cards can in no way affect the sorted order. If a print of the resulting sorted deck is requested, it is performed during this pass.

During any sort collation the data cards are ordered by comparing half-fields from left to right. Each bulk data card may contain up to ten, eight column (character) fields.

Because of computer word size constraints, each data card is stored into twenty memory words, four characters (half a card field) per word. Sorting proceeds by comparing the first words (4 characters) from each card. If an order cannot be established, i.e., the words are equal, the second words from each card are compared, and so on, until an order is established or total duplication is determined. Each field (8 characters) is left (BCD) or right (integer) justified prior to sorting to eliminate leading or trailing blanks. The characters within the first field of each card are converted to a special internal character set prior to comparing to eliminate machine dependent collation sequences which might order the same cards differently on different machines. This internal set forces the collation order to be ascending from blank through all numbers then all letters. A flowchart is given in Figure 1.

4.4.5 Subroutines

In the following, note that XRECPS, RPAGE, INITCØ, XFADJ, XBCDBI, XPRETY, CRDFLG, EXTINT, and INTEXT are secondary entry points in XRECPS.

4.4.5.1 Subroutine Name: XRECPS

1. Entry Point: XRECPS

2. Purpose: Positions the continuation card file to the proper record (card image) as determined from the in-core continuation card dictionary.

3. Calling Sequence: CALL XRECPS (NEW, ØLD)

Where: NEW = the file position being requested

ØLD = the file position last requested.

Both arguments are integer record numbers.

4.4.5.2 Subroutine Name: RPAGE

1. Entry Point: RPAGE

2. Purpose: Counts output print lines for XSØRT, and performs the necessary interface with the system subroutine PAGE.

3. Calling Sequence: CALL RPAGE (NLINE)

Where: NLINE = integer number of lines being output. If NLINE $\geq$ 100 a page eject is

forced and the line count is set to NLINE - 100.

4.4.5.3 Subroutine Name: INITCØ

1. Entry Point: INITCØ

2. Purpose: Initializes machine dependent masks and constants within XSØRT.

3. Calling Sequence: CALL INITCØ

4.4.5.4 Subroutine Name: XFADJ

1. Entry Point: XFADJ

2. Purpose: Adjusts four character fields, left or right, two or four fields at a time. If the fields contain only integers, the shift is right, otherwise the shift will be left. This routine determines only the direction of shift required. Actual shifting is performed by XFADJ1.

3. Calling Sequence: CALL XFADJ (BUF,SD,K)

Where: BUF = field array to be shifted

SD = { 0, shift two fields at a time
     { 1, shift four fields at a time

K = { 0, returned if right shift was done.
    { 1, returned if left shift was done.

4.4.5.5 Subroutine Name: XBCDBI

1. Entry Point: XBCDBI

2. Purpose: Converts two, four character BCD integer fields (right adjusted in the left most four characters of the computer word) into a single binary integer (right adjusted in the second of the two input words).

3. Calling Sequence: CALL XBCDBI (BUF)

Where: BUF = two word array to be converted.

4.4.5.6 Subroutine Name: XPRETY

1. Entry Point: XPRETY

2. Purpose: "Pretties-up" printed output by left adjusting all fields to eliminate any leading zeros introduced when integer fields are right adjusted.

3. Calling Sequence: CALL XPRETY (BUF)

Where: BUF = card image array.

4.4.5.7 Subroutine Name: CRDFLG

1. Entry Point: CRDFLG

2. Purpose: Sets the card type flags within the restart tables.

3. Calling Sequence: CALL CRDFLG (CARD)

Where: CARD = first of two word card type field.

4.4.5.8 Subroutine Name: EXTINT

1. Entry Point: EXTINT

2. Purpose: Converts card type field from the machine dependent character code to an internal machine independent code.

3. Calling Sequence: CALL EXTINT (CTYBF)

Where: CTYBF = first of two word card type field.

4.4.5.9 Subroutine Name: INTEXT

1. Entry Point: INTEXT

2. Purpose: Converts the card type field from an internal machine independent code to the machine dependent character code.

3. Calling Sequence: CALL INTEXT (CTYBF)

Where: CTYBF = first of two word card type field.

4.4.5.10 Subroutine Name: XFADJ1

1. Entry Point: XFADJ1

2. Purpose: Adjust four character fields left or right, two or four fields at a time. This routine performs actual shifting with the direction of shift controlled through the

calling sequence. (Note entry point XFADJ).

3. Calling Sequence: CALL XFADJ1 (BUF,SHIFT,SD)

Where: BUF = Field Array to be shifted.

SHIFT = Function LSHIFT or RSHIFT.

SD = {0, shift two fields at a time.
{1, shift four fields at a time.

4.4.5.11 Function Name: ISFT

1. Entry Point: ISFT

2. Purpose: Performs special shifting functions for subroutine XFAJ1.

3. Calling Sequence: CALL ISFT

RESULT = ISFT(BUF,SFTCNT,J)

where: BUF = Word to be shifted.

SFTCNT = Bits to be shifted.

J = Shift direction control; 3 = right, 4 = left.

4.4.6 Design Requirements

1. Data cards operated upon by XSØRT must conform to the NASTRAN format for bulk data cards (ten, eight character fields per card). See section 2 of the User's Manual for details.

2. Data cards must contain only valid BCD key punch codes or blanks. Non-standard multi-punched code (e.g., some IBM EBCDIC) will cause unpredictable results.

3. XSØRT requires sufficient open core to contain five GINØ buffers and a work buffer for at least ten data cards. (Each data card requires twenty-one core locations). Sort efficiency increases in proportion to the size of the work buffer.

4. The continuation card dictionary must fit into the core work buffer during the final pass. Each continuation card requires two dictionary locations.

5. XSØRT logic is biased toward input that is already sorted. That is, the program will operate at a much greater speed if verifying a sort rather than producing a sort.

4.4.7 Diagnostic Messages

XSØRT can produce two catagories of diagnostic messages. The first are termed USER messages and deal with bulk data card errors. The second are termed SYSTEM messages which are generally fatal in nature and indicate serious I/Ø malfunctions.

XSØRT message numbers includes 201 through 216. All messages are listed and explained in section 6 of the User's Manual.

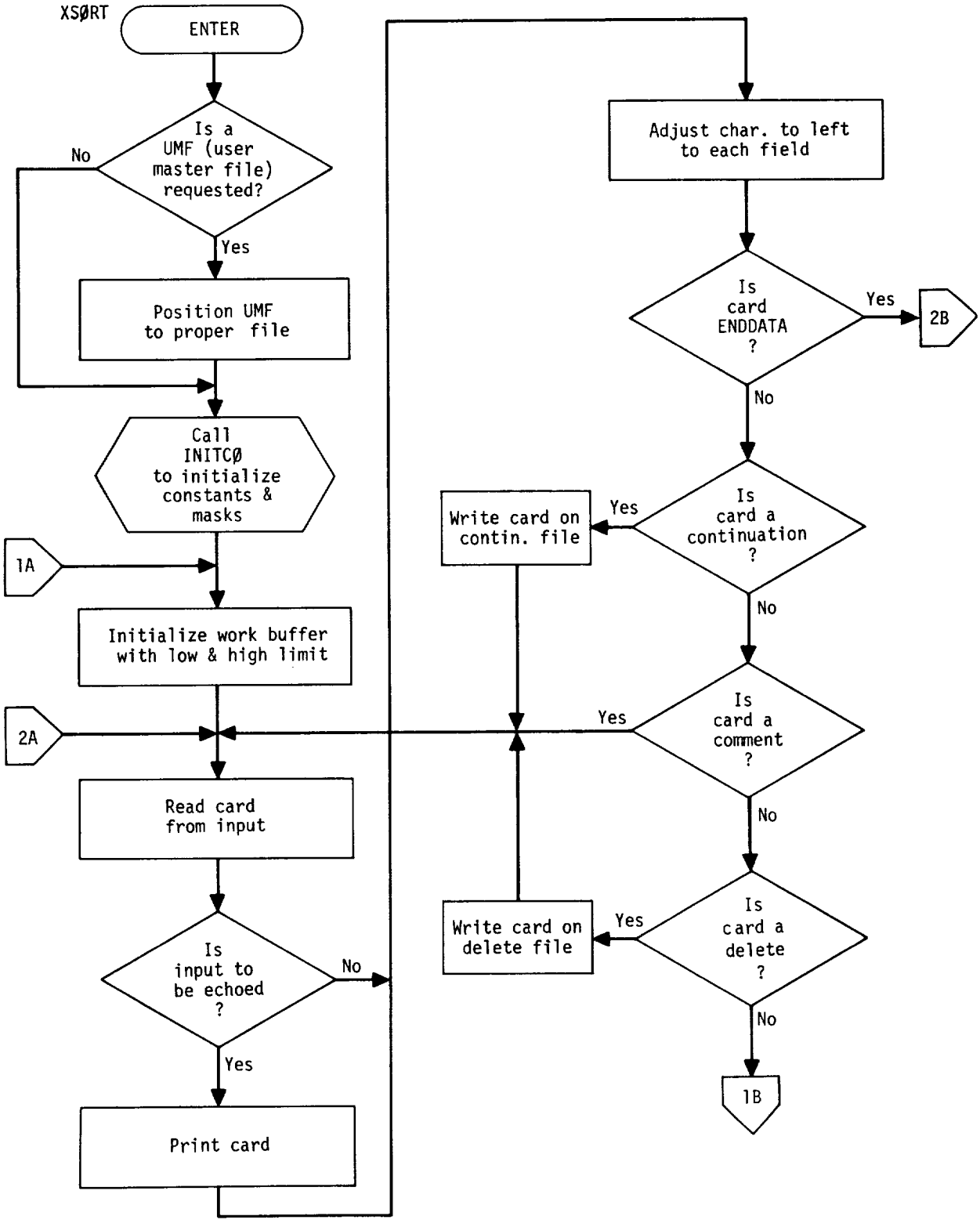


Figure 1.(a) Flowchart for module XSØRT.

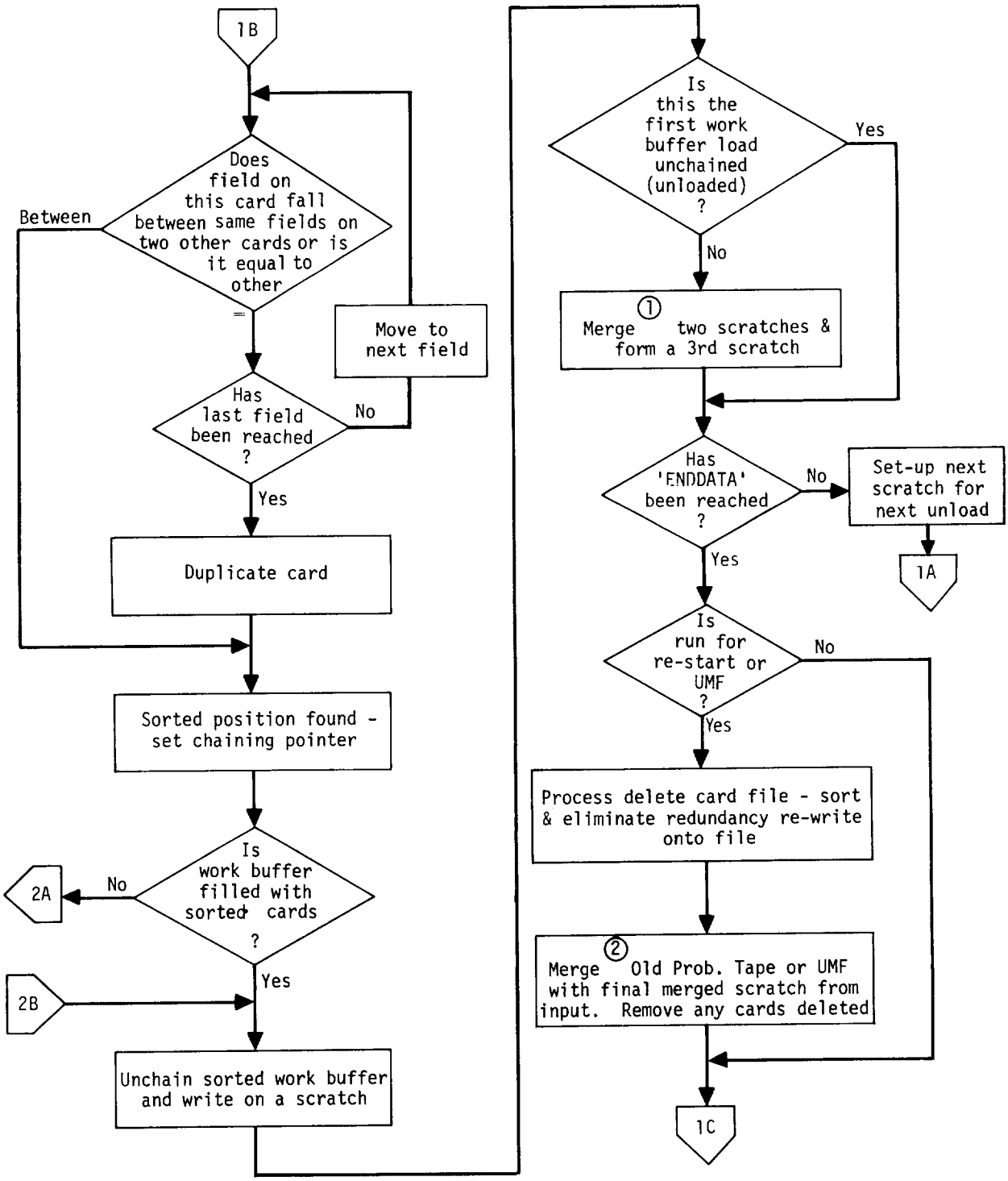


Figure 1.(b) Flowchart for module XSØRT.

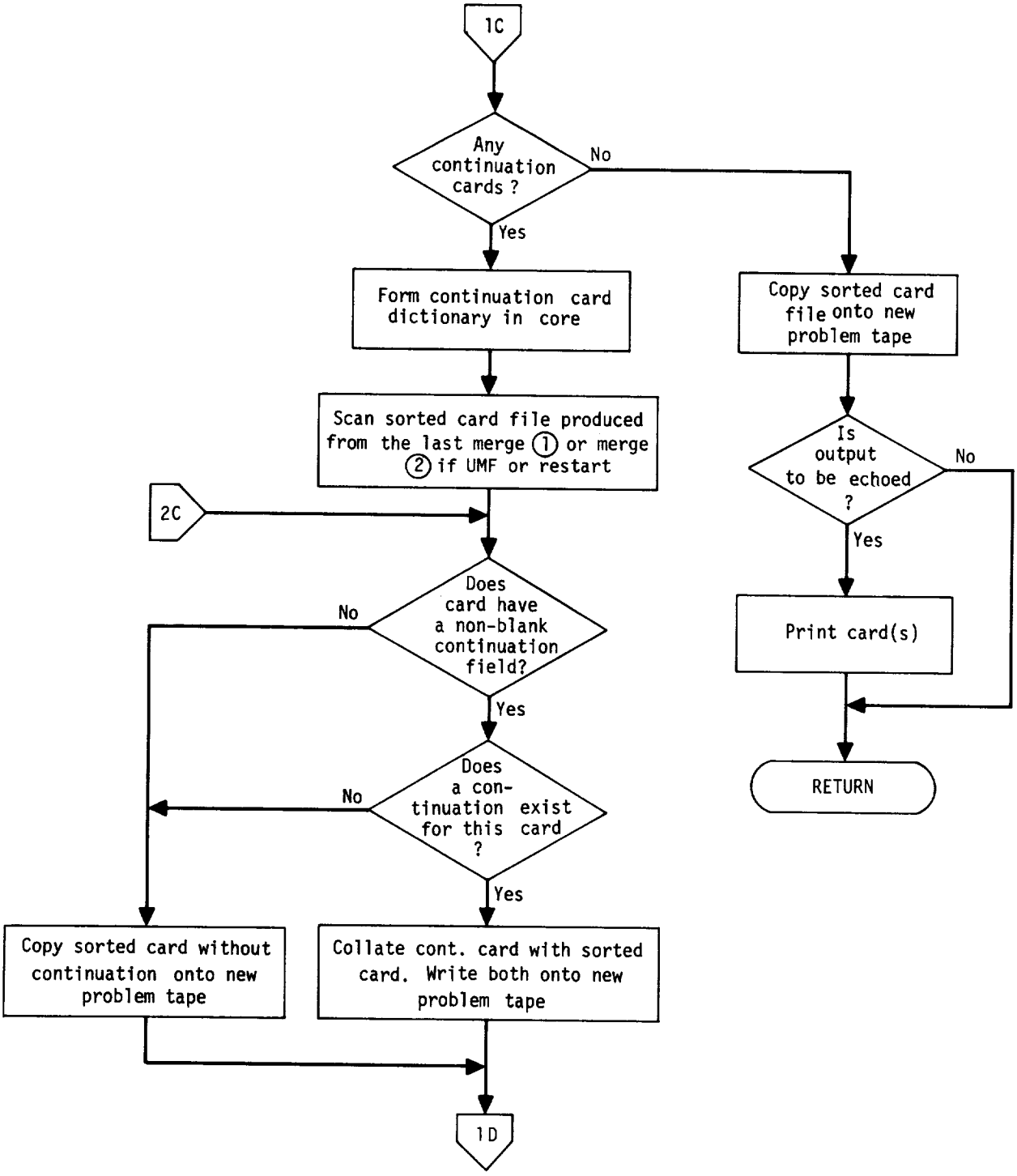


Figure 1.(c) Flowchart for module XSØRT.

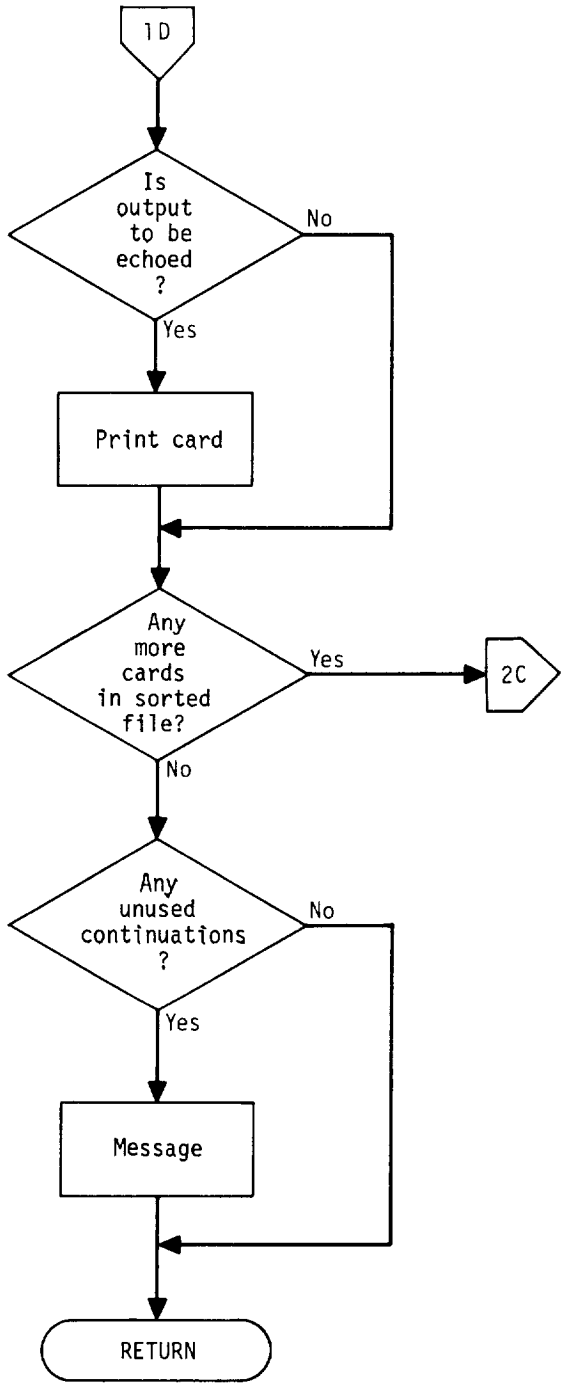


Figure 1.(d) Flowchart for module XSØRT.

4.5 EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

4.5.1 Entry Point: IFP

4.5.2 Purpose

To process the Bulk Data Deck sorted by Executive Preface module, XSØRT. This task is accomplished as follows: 1) the sorted Bulk Data Deck is read from the New Problem Tape (NPTP) card-by-card; 2) the contents of each field of each card are validated (see section 2.4 of the User's Manual for detailed descriptions of each bulk data card); 3) card images or modified card images are written on data blocks or the NPTP or the Data Pool File (see section 2.3.2 of the Programmer's Manual for details on the formats of these data blocks).

4.5.3 Calling Sequence

CALL IFP. IFP, an Executive Preface module, is called only by the Preface driver, subroutine SEMINT.

4.5.4 Input

The input to IFP consists of the Bulk Data Deck sorted by Executive Preface module XSØRT.

4.5.5 Output

The output of IFP consists of: 1) data blocks used in Rigid Formats; 2) the AXIC data block, which is processed by Executive Preface module IFP3 and is present only if the NASTRAN run is a conical shell (a unique structural element) problem solution; 3) the PVT Executive table, which contains the names and values of all DMAP parameters input by means of the PARAM bulk data card, and which is written on the New Problem Tape to be processed by Executive Preface module XGPI; and 4) DMI's (Direct Matrix Inputs) and DTI's (Direct Table Inputs), each of which is written on the Data Pool File as a data block and is indistinguishable from any matrix data block or table data block pooled by the Executive Segment File Allocator (XSFA) module.

4.5.5.1 Output Data Blocks Used in Rigid Formats

GEØM1 - Grid point, coordinate system, and sequence data.

GEØM2 - Element connection data.

GEØM3 - Static loads and temperature data.

GEØM4 - Displacement set definitions data.

EPT - Element Property Table.

MPT - Material Property Table.

DIT - Direct Input Tables.

EDT - Element Deformation Table.

DYNAMICS - Collection of bulk data cards for a dynamics problem.

MATPØØL - Data block containing matrices input on DMIG bulk data cards.

Note: Do not confuse the DTI (Direct Table Input) bulk data card and the DIT (Direct Input Table) data block.

4.5.6 Method

4.5.6.1 General Comments

The bulk data cards processed by IFP are classified into five categories. Listed below is a brief explanation of each with a few examples.

1. Closed End Cards (Fixed Length Card)

Cards such as CQUAD2 and PRØD go through all the standard bulk data card checks (see 4.5.6.2) before being processed by the card dependent subroutines within IFP, (IFSiP, i = 1, 2, 3, 4). These closed end cards are output to one of the standard GINØ files.

2. Open Ended Cards (Variable Length Cards)

In cards such as SPC1 and PLFACT, since the length and therefore the formats are not known, the bulk data checks using the data initialized in the block data subprograms must be made in the card dependent subroutines. Also, since the length is not known, a flag is placed at the end of the information for that card before being written on the file in order that routines reading an open ended card will be signaled as to an end-of-card condition.

3. GRDSET and BARØR Cards

Special cards such as GRDSET and BARØR are not output to a GINØ data block, but are stored in local variables, and provide default values for the GRID and CBAR cards.

4. DMI and DTI Cards

DMI and DTI cards are unique in the manner in which they are used by the user and processed by IFP. The DMI card enables the user to define matrix data blocks directly, while the DTI card gives the user the capability to input his own table data blocks directly. The user must write a DMAP sequence or use the ALTER feature - see section 2.2 of the User's Manual - in the Executive Control Deck to alter the Rigid Format chosen in order to use the DMI or DTI feature since he is defining his own data blocks. Both DMI and DTI cards are written directly onto the Data Pool File.

5. PARAM Cards

PARAM bulk data cards are stored in open core by IFP until the entire Bulk Data Deck has been processed. PARAM cards are then written as the PVT (Parameter Value Table) on the NPTP for subsequent processing by the Executive Preface module XGPI.

4.5.6.2 Card Processing

1. IFP searches the NPTP for the Bulk Data Deck and extracts it in 20 word (one physical card) segments. Each card is passed to subroutine RCARD, which takes the BCD card images and converts the fields thereon to values, and identifies the values as to type: blank data field, integer data field, real single precision data field, BCD data field, real double precision data field or a data field which is in error.

IFP always has two physical bulk data cards in internal storage areas: the "current" card and the "next" card. M is the local FØRTRAN array where the values of the current bulk data card are located, and M1 the local FØRTRAN array where the values of the next card are located. After the current card is processed, the data in M1 are transferred into M, and new card values for M1 are input from the NPTP. When the values from M1 are transferred to M, the first two words (the card mnemonic) are stripped off. M1(3) is stored in M(1), M1(4) is stored in M(2), and so on.

2. /IFPX1/ is referenced to verify the admissibility of the name (mnemonic) of the particular card taken from the NPTP.

3. The approach acceptability flag is checked. This flag is defined as follows:

0 = OK for any approach;

1 = Not used by displacement approach;

2 = Illegal for displacement approach.

The approach flag (DISPL, DMAP) is found in /SYSTEM/.

4. The proper output files are established. See Table 1 or Table 2 for the output file on which the various bulk data cards will reside.

5. Uniqueness flags, which reside in CØMMØN/IFPX5/, are defined for each card type as follows:

0 - No check is made;

1 - A check is made;

2 - A special check is made.

For example, on the bulk data card CØNRØD, field 2 is the EID, and it must be unique with respect to all other CØNRØD EID's.

6. The next physical card is read from the NPTP. This will be the next card to be processed.

7. A check for too many continuation cards is made. This check is made on fixed length cards only.

8. A check is made for the minimum and maximum number of words for a logical bulk data card.

9. A check for the proper types of values for the fields on a card is made by referencing /IFPX7/, which contains format codes for each card type as follows:

0 = Blank

1 = Integer

2 = Real

3 = BCD

4 = Double Precision

5 = Anything.

When RCARD passes format values to IFP, a format code of 0 will override this check. In the card dependent code check (step 10), the value will be looked at to see if it is in error.

10. An auxiliary subroutine IFSiP, i = 1, 2, 3, 4, is called to execute card dependent code.

11. If the input card passes the tests in the card dependent code, the data are written on the appropriate GINØ output file.

4.5.6.3 Module Conclusion

When the sorted Bulk Data Deck has been exhausted, the following steps are carried out.

1. The appropriate trailer codes are written for each data block. For listings of trailer information reference section 2.3.2.

2. The PVT is written on the NPTP.

3. Restart flags are set in /IFPXO/.

4.5.7 Subroutines

IFP uses the utility routine RCARD described in section 3.4.20.

4.5.7.1 Block Data Subprogram: IFX1BD

Purpose: To initialize /IFPX1/, which is used by IFP to validate card names. All bulk data card names must appear in this table.

4.5.7.2 Block Data Subprogram: IFX2BD

Purpose: To initialize /IFPX2/. This table contains two words per entry (two words per card type): the first gives the GINØ output file number, and the second gives the approach acceptability flag.

4.5.7.3 Block Data Subprogram: IFX3BD

Purpose: To initialize /IFPX3/. This table contains two words per entry (two words per card type): the first word is used as the Conical Shell Problem flag, and the second word is used internally to store the number of words to be output to the GINØ output file.

4.5.7.4 Block Data Subprogram: IFX4BD

Purpose: To initialize /IFPX4/. This table contains two words per entry (two words per card type): the first is the minimum number of words allowable, the second is the maximum number of words allowable. The first word of an entry being negative implies the card is open ended.

4.5.7.5 Block Data Subprogram: IFX5BD

Purpose: To initialize /IFPX5/. This table contains two words per entry (two words per card type): the first is a pointer into /IFPX7/, the second is the field 2 uniqueness check flag.

4.5.7.6 Block Data Subprogram: IFX6BD

Purpose: To initialize /IFPX6/. This table contains two words per entry (two words per card type): the first is header word 1 (card type), the second is header word 2 (trailer bit position) of the three word header information of each logical record, which corresponds to all the data of particular bulk data card type. See section 2.3.2 for details.

4.5.7.7 Block Data Subprogram: IFX7BD

Purpose: To initialize /IFPX7/. Each entry contains the admissible sequence of format codes for that card type (see step 9 in section 4.5.6.2 above).

4.5.7.8 Subroutine Name: IFSiP, i = 1, 2, 3, 4

1. Entry Point: IFSiP, i = 1, 2, 3, 4

2. Purpose: These are the four subroutines that the module driver IFP calls to execute card dependent code.

3. Calling Sequence: CALL IFSiP (n_1,n_2,n_3)

   n_1 - FØRTRAN statement number defining the return taken in the event of a format or data error.

   n_2 - FØRTRAN statement number defining the return taken when local variables are set to provide default values for appropriate cards.

   n_3 - FØRTRAN statement number defining the return taken in the event of a data error.

4.5.7.9 Subroutine Name: IFPPK

1. Entry Points: IFPPK,IFPPKI,IFPPKN

2. Purpose: To provide IFP with a shortened version of subroutine BLDPK in order to preserve core storage on machines with minimal memory capacities. This routine is used to write DMI cards on the Data Pool File in NASTRAN matrix format.

4.5.8 Design Requirements

Open core is defined in /IFPXX/. Open core is used to store all PARAM cards until the Bulk Data Deck has been exhausted, at which time the PARAM cards are written on the NPTP as the PVT Executive table.

4.5.9 Diagnostic Messages

If a fatal error is detected during any phase of the processing of module IFP, the NØGØ flag will be set, and the error message will be printed out. IFP will continue processing data cards until all are processed. IFP may issue error messages 301 through 332.

Table 1(a). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

The following list gives an explanation of the column headings on the following pages of Table 1.

A = Internal IFP Bulk Data Card Number

B = Bulk Data Card Name (an asterisk following a name implies the card is not available)

C = Internal IFP GINØ Output File Number

D = Data Block Name

E = Approach Acceptance Indicator

-2 = Illegal for the Force Approach

-1 = Not Used by the Force Approach

0 = OK for any Approach

1 = Not Used by the Displacement Approach

2 = Illegal for the Displacement Approach

F = Minimum Number of Words Allowed Per Logical Card (F negative implies an open ended card)

G = Maximum Number of Words Allowed Per Logical Card

H = Format Check Pointer Into IFX7BD

I = Field 2 Uniqueness Check Flag

0 = No Check is Made

1 = Check is Made

2 = Special

J = Subroutine LØCATE Code for Card on Output Data Block

K = Trailer Bit Position

L = Pointer to Secondary (Card Dependent) Code

S1 = Subroutine IFS1P

S2 = Subroutine IFS2P

S3 = Subroutine IFS3P

S4 = Subroutine IFS4P

M = FØRTRAN Statement Number in the Card Dependent Subroutines

N = Conical Shell Problem Flag

-1 = Illegal for Shell Mode

Table 1(b). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

0 = OK for Shell Mode

1 = Puts Card Into Different Data Block

0 = Users Map for Data Blocks IFX2BD,...,IFX6BD

Values for I = 1,2 or 3

J = 1,2 or 3

H = A,B,C,D or E

K = 1,2,3,4,5 or 6

I = Is Data Statement in the Block Data Program

J = The Group of A Through E Continuation Card Blocks Within the Ith Data Statement

H = Alphabetic Character in Col 6 (Continuation Column) in the Jth Group

K = The Pair Number on Line H Where the Actual Data is

Table 1(c). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
1	GRID	1	GEØM1	0	4	12	1	1	4501	45	S3	100	-1	11A1
2	GRDSET	1	GEØM1	0	4	12	13	2	0	0	S3	200	-1	11A2
3	BEAMØR*	8	GEØM2	-2	4	12	25	2	0	0	S3	300	-1	11A3
4	SEQGP	1	GEØM1	-1	4	8	37	0	5301	53	S1	40	-1	11A4
5	CØRD1R	1	GEØM1	0	4	8	37	0	1801	18	S1	500	-1	11A5
6	CØRD1C	1	GEØM1	0	4	8	37	0	1701	17	S1	600	-1	11A6
7	CØRD1S	1	GEØM1	0	4	8	37	0	1901	19	S1	700	-1	11B1
8	CØRD2R	1	GEØM1	0	12	16	45	1	2101	21	S1	800	-1	11B2
9	CØRD2C	1	GEØM1	0	12	16	45	1	2001	20	S1	900	-1	11B3
10	CØRD2S	1	GEØM1	0	12	16	45	1	2201	22	S1	1000	-1	11B4
11	PLØTEL	8	GEØM2	0	4	8	505	0	5201	52	S1	1111	-1	11B5
12	SPC1	10	GEØM4	-2	4	8	-1	0	5481	58	S3	3980	-1	11B6
13	SPCADD	10	GEØM4	-2	4	8	-1	1	5491	59	S3	4020	1	11C1
14	SUPØRT	10	GEØM4	-2	4	8	37	0	5601	56	S1	1400	-1	11C2
15	ØMIT	10	GEØM4	-2	4	8	37	0	5001	50	S1	1400	-1	11C3
16	SPC	10	GEØM4	-2	4	8	101	0	5501	55	S1	1600	-1	11C4
17	MPC	10	GEØM4	-2	4	8	-1	0	4901	49	S3	1700	-1	11C5
18	FØRCE	9	GEØM3	0	8	12	109	0	4201	42	S1	1800	1	11C6
19	MØMENT	9	GEØM3	0	8	12	109	0	4801	48	S1	1800	1	11D1
20	FØRCE1	9	GEØM3	0	8	12	121	0	4001	40	S1	2000	-1	11D2
21	MØMENT1	9	GEØM3	0	8	12	121	0	4601	46	S1	2000	-1	11D3
22	FØRCE2	9	GEØM3	0	8	12	133	0	4101	41	S1	2200	-1	11D4
23	MØMENT2	9	GEØM3	0	8	12	133	0	4701	47	S1	2200	-1	11D5
24	PLØAD	9	GEØM3	0	8	12	145	0	5101	51	S1	2400	-1	11D6
25	SLØAD	9	GEØM3	-2	4	8	157	0	5401	54	S1	2500	-1	11E1
26	GRAV	9	GEØM3	0	8	12	165	1	4401	44	S1	2600	1	11E2
27	TEMP	9	GEØM3	0	4	8	157	0	5701	57	S1	2500	-1	11E3
28	GENEL	8	GEØM2	-2	-4	9	-1	1	4301	43	S3	2800	-1	11E4
29	PRØD	2	EPT	0	4	12	165	1	902	9	S1	2900	-1	11E5
30	PTUBE	2	EPT	0	4	12	177	1	1602	16	S1	2920	-1	11E6
31	PVISC	2	EPT	-2	4	8	189	0	1802	18	S1	310	-1	12A1
32	PBEAM*	2	EPT	0	8	24	197	1	102	1	S1	2960	-1	12A2
33	PTRIA1	2	EPT	0	4	16	221	1	1202	12	S1	2980	-1	12A3
34	PTRIA2	2	EPT	0	4	8	237	0	1302	13	S1	3000	-1	12A4
35	PTRBSC	2	EPT	0	4	12	257	1	1102	11	S1	3020	-1	12A5
36	PTRPLT	2	EPT	0	4	12	257	1	1502	15	S1	3020	-1	12A6
37	PTRMEM	2	EPT	0	4	8	237	0	1402	14	S1	3000	-1	12B1
38	PQUAD1	2	EPT	0	4	16	221	0	702	7	S1	2980	-1	12B2
39	PQUAD2	2	EPT	0	4	8	237	0	802	8	S1	3000	-1	12B3
40	PQDPLT	2	EPT	0	4	12	257	0	602	6	S1	3020	-1	12B4
41	PQDMEM	2	EPT	0	4	8	237	0	502	5	S1	3000	-1	12B5
42	PSHEAR	2	EPT	0	4	8	237	0	1002	10	S1	3000	-1	12B6
43	PTWIST	2	EPT	0	4	8	237	0	1702	17	S1	3000	-1	12C1
44	PMASS	2	EPT	-2	4	8	269	0	402	4	S1	3200	-1	12C2
45	PDAMP	2	EPT	-2	4	8	269	0	202	2	S1	3200	-1	12C3
46	PELAS	2	EPT	-2	4	8	497	0	302	3	S1	3240	-1	12C4
47	CØNRØD	8	GEØM2	0	8	12	277	1	1601	16	S1	3260	-1	12C5
48	CRØD	8	GEØM2	0	4	8	37	0	3001	30	S1	3281	-1	12C6
49	CTUBE	8	GEØM2	0	4	8	37	0	3701	37	S1	3282	-1	12D1
50	CVISC	8	GEØM2	-2	4	8	37	0	3901	39	S1	3283	-1	12D2
51	CBEAM*	8	GEØM2	-2	8	24	289	1	101	1	S3	3340	-1	12D3
52	CTRIA1	8	GEØM2	0	8	12	313	1	3301	33	S1	3360	-1	12D4
53	CTRIA2	8	GEØM2	0	8	12	313	1	3401	34	S1	3360	-1	12D5
54	CTRBSC	8	GEØM2	0	8	12	313	1	3201	32	S1	3360	-1	12D6
55	CTRPLT	8	GEØM2	0	8	12	313	1	3601	36	S1	3360	-1	12E1
56	CTRMEM	8	GEØM2	0	8	12	313	1	3501	35	S1	3360	-1	12E2
57	CQUAD1	8	GEØM2	0	8	12	325	1	2801	28	S1	3460	-1	12E3
58	CQUAD2	8	GEØM2	0	8	12	325	1	2901	29	S1	3460	-1	12E4

Table 1(d). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
59	CQDPLT	8	GEØM2	0	8	12	325	1	2701	27	S1	3460	-1	12E5
60	CQDMEM	8	GEØM2	0	8	12	325	1	2601	26	S1	3460	-1	12E6
61	CSHEAR	8	GEØM2	0	8	12	337	1	3101	31	S1	3540	-1	13A1
62	CTWIST	8	GEØM2	0	8	12	337	1	3801	38	S1	3540	-1	13A2
63	CØNM1	8	GEØM2	0	8	28	349	1	1401	14	S1	3580	-1	13A3
64	CØNM2	8	GEØM2	0	8	20	377	1	1501	15	S1	3600	-1	13A4
65	CMASS1	8	GEØM2	-2	4	12	337	1	1001	10	S1	3620	-1	13A5
66	CMASS2	8	GEØM2	-2	4	12	397	1	1101	11	S1	3623	-1	13A6
67	CMASS3	8	GEØM2	-2	4	8	37	0	1201	12	S1	3674	-1	13B1
68	CMASS4	8	GEØM2	-2	4	8	409	0	1301	13	S1	3697	-1	13B2
69	CDAMP1	8	GEØM2	-2	4	12	337	1	201	2	S1	3620	-1	13B3
70	CDAMP2	8	GEØM2	-2	4	12	397	1	301	3	S1	3623	-1	13B4
71	CDAMP3	8	GEØM2	-2	4	8	37	0	401	4	S1	3675	-1	13B5
72	CDAMP4	8	GEØM2	-2	4	8	409	0	501	5	S1	3698	-1	13B6
73	CELAS1	8	GEØM2	-2	4	12	337	1	601	6	S1	3620	-1	13C1
74	CELAS2	8	GEØM2	-2	4	12	417	1	701	7	S1	3800	-1	13C2
75	CELAS3	8	GEØM2	-2	4	8	37	0	801	8	S1	3676	-1	13C3
76	CELAS4	8	GEØM2	-2	4	8	409	0	901	9	S1	3699	-1	13C4
77	MAT1	3	MPT	0	4	20	429	1	103	1	S1	3860	0	13C5
78	MAT2	3	MPT	0	8	20	449	1	203	2	S1	3880	-1	13C6
79	CTRIARG	8	GEØM2	-2	8	12	738	1	1708	17	S4	790	-1	13D1
80	CTRAPRG	8	GEØM2	-2	8	12	737	1	1808	18	S4	800	-1	13D2
81	DEFØRM	4	EDT	-2	4	8	157	0	104	1	S1	2500	-1	13D3
82	PARAM	6	PVT	0	-5	16	-1	2	0	0	S3	3960	0	13D4
83	MPCADD	10	GEØM4	-2	4	8	-1	1	4891	60	S3	4020	1	13D5
84	LØAD	9	GEØM3	0	4	8	-1	1	4551	61	S3	4060	1	13D6
85	EIGR	7	DYNAMICS	-2	14	18	469	1	307	3	S2	850	0	13E1
86	EIGB	7	DYNAMICS	-2	14	18	469	1	107	1	S2	850	0	13E2
87	EIGC	7	DYNAMICS	-2	-4	10	-1	1	207	2	S2	870	0	13E3
88	REACT	8	GEØM2	2	8	24	597	1	5251	62	S4	880	-1	13E4
89		1		0	8	8	-1	2			S2	890	0	13E5
90	MATS1	3	MPT	-2	4	16	545	1	503	5	S4	900	-1	13E6
91	MATT1	3	MPT	0	4	16	545	1	703	7	S4	900	0	21A1
92	ØMIT1	10	GEØM4	-2	-4	8	-1	0	4951	63	S3	3981	-1	21A2
93	TABLEM1	5	DIT	0	-4	16	-1	1	105	1	S2	930	0	21A3
94	TABLEM2	5	DIT	0	-4	16	-1	1	205	2	S2	930	0	21A4
95	TABLEM3	5	DIT	0	-4	16	-1	1	305	3	S2	930	0	21A5
96	TABLEM4	5	DIT	0	-4	16	-1	1	405	4	S2	960	0	21A6
97	TABLES1	5	DIT	-2	-4	16	-1	1	3105	31	S2	930	-1	21B1
98	TEMPD	9	GEØM3	0	4	12	269	0	5641	65	S4	980	-1	21B2
99	TABLES2*	5	DIT	-2	-4	16	-1	1	3205	32	S2	930	-1	21B3
100	TABLES3*	5	DIT	-2	-4	16	-1	1	3305	33	S2	930	-1	21B4
101	TABLES4*	5	DIT	-2	-4	16	-1	1	3405	34	S2	960	-1	21B5
102	MATT2	3	MPT	0	4	16	525	1	803	8	S4	1020	-1	21B6
103	MATS2*	3	MPT	-2	4	16	525	1	603	6	S4	1020	-1	21C1
104	CTØRDRG	8	GEØM2	-2	4	12	750	1	1908	19	S4	1040	-1	21C2
105	SPØINT	8	GEØM2	-2	4	9	794	0	5551	49	S4	1050	-1	21C3
106	SEQD	8	GEØM2	2	-4	9	-1	0	7508	75	S2	1060	-1	21C4
107	SEQDBFE	8	GEØM2	2	4	8	37	0	4608	46	S2	1070	-1	21C5
108	QDSEP	8	GEØM2	2	4	8	37	0	4808	48	S4	1080	-1	21C6
109	SPQUAD1	2	EPT	2	4	16	221	0	6502	65	S1	2980	-1	21D1
110	SPQUAD2	2	EPT	2	4	8	237	0	6602	66	S1	3000	-1	21D2
111	SPQDMEM	2	EPT	2	4	8	237	0	6302	63	S1	3000	-1	21D3
112	SPQDPLT	2	EPT	2	4	12	257	0	6402	64	S1	3020	-1	21D4
113	ZI	13	FØRCE	2	8	12	485	0	8913	89	S4	1130	-1	21D5
114	CTRIA3	8	GEØM2	2	8	12	313	1	3458	41	S1	3360	-1	21D6
115	PTRIA3	2	EPT	2	-4	8	-1	1	1352	24	S1	1690	-1	21E1
116	SETRBFE	8	GEØM2	2	4	8	549	0	4708	47	S1	1160	-1	21E2

Table 1(e). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
117	VECDN	13	FØRCE	2	8	12	177	1	8313	83	S4	1170	-1	21E3
118	VECGP	13	FØRCE	2	8	12	585	1	8413	84	S4	1180	-1	21E4
119	DMI	12	PØØL	0	-4	16	-1	0	0	0	S2	1190	0	21E5
120	DMIG	14	MATPØØL	-2	-4	12	-1	0	114	1	S2	1200	0	21E6
121	PTØRDRG	2	EPT	-2	4	8	237	0	2102	21	S1	3000	-1	22A1
122	MAT3	3	MPT	-2	4	20	449	1	1403	14	S4	1220	-1	22A2
123	DLØAD	7	DYNAMICS	-2	4	8	-1	1	57	5	S3	4060	0	22A3
124	EPØINT	7	DYNAMICS	-2	4	9	794	0	707	7	S4	1050	0	22A4
125	FREQ1	7	DYNAMICS	-2	4	8	705	0	1007	10	S1	1250	0	22A5
126	FREQ	7	DYNAMICS	-2	4	88	-1	1	1307	13	S3	1260	0	22A6
127	NØLIN1	7	DYNAMICS	-2	8	12	725	0	3107	31	S1	1270	0	22B1
128	NØLIN2	7	DYNAMICS	-2	8	12	725	0	3207	32	S1	1280	0	22B2
129	NØLIN3	7	DYNAMICS	-2	8	12	725	0	3307	33	S1	1290	0	22B3
130	NØLIN4	7	DYNAMICS	-2	8	12	725	0	3407	34	S1	1290	0	22B4
131	RLØAD1	7	DYNAMICS	-2	8	8	337	1	5107	51	S3	1310	0	22B5
132	RLØAD2	7	DYNAMICS	-2	8	8	337	1	5207	52	S3	1310	0	22B6
133	TABLED1	5	DIT	-2	-4	16	-1	1	1105	11	S2	930	0	22C1
134	TABLED2	5	DIT	-2	-4	16	-1	1	1205	12	S2	930	0	22C2
135	SEQEP	7	DYNAMICS	-1	4	8	37	0	5707	57	S1	40	-1	22C3
136	TF	7	DYNAMICS	-2	8	12	-1	0	6207	62	S1	1360	0	22C4
137	TIC	7	DYNAMICS	-2	4	12	713	0	6607	66	S1	1370	0	22C5
138	TLØAD1	7	DYNAMICS	-2	8	8	681	1	7107	71	S3	1380	0	22C6
139	TLØAD2	7	DYNAMICS	-2	8	16	689	1	7207	72	S3	1390	0	22D1
140	TABLED3	5	DIT	-2	-4	16	-1	1	1305	13	S2	930	0	22D2
141	TABLED4	5	DIT	-2	-4	16	-1	1	1405	14	S2	960	0	22D3
142	TSTEP	7	DYNAMICS	-2	4	8	-1	1	8307	83	S1	1420	0	22D4
143	DSFACT	3	MPT	-2	4	8	-1	1	53	10	S3	1430	0	22D5
144	AXIC	15	AXIC	-2	4	8	93	0	515	5	S3	1440	0	22D6
145	RINGAX	15	AXIC	-2	4	12	245	1	5615	56	S3	1450	0	22E1
146	CCØNEAX	15	AXIC	-2	4	8	645	1	2315	23	S3	1460	0	22E2
147	PCØNEAX	2	EPT	-2	4	28	653	1	152	19	S3	1470	0	22E3
148	SPCAX	15	AXIC	-2	4	12	485	0	6215	62	S3	1480	0	22E4
149	MPCAX	15	AXIC	-2	4	8	-1	0	4015	40	S3	1490	0	22E6
150	ØMITAX	15	AXIC	-2	4	8	337	0	4315	43	S3	1500	0	22E6
151	SUPAX	15	AXIC	-2	4	8	337	0	6415	64	S3	1500	0	23A1
152	PØINTAX	15	AXIC	-2	4	8	517	1	4915	49	S3	1520	0	23A2
153	SECTAX	15	AXIC	-2	4	12	177	1	6015	60	S3	1530	0	23A3
154	PRESAX	15	AXIC	-2	4	12	61	0	5215	52	S3	1540	0	23A4
155	TEMPAX	15	AXIC	-2	4	8	237	0	6815	68	S3	1550	0	23A5
156	FØRCEAX	15	AXIC	-2	4	12	109	0	2115	21	S3	1560	0	23A6
157	MØMAX	15	AXIC	-2	4	12	109	0	3815	38	S3	1560	0	23B1
158	EIGP	7	DYNAMICS	-2	4	8	561	0	257	4	S1	1580	0	23B2
159	MASSC	13	FØRCE	2	4	8	-1	0	3313	33	S3	1590	-1	23B3
160	EDFIR	13	FØRCE	2	4	8	37	0	1513	15	S3	1600	-1	23B4
161	DFØRM	4	EDT	2	4	8	101	0	204	2	S3	1610	-1	23B5
162	TABDMP1	5	DIT	-2	-4	16	-1	1	15	21	S2	930	0	23B6
163	TABDMP2*	5	DIT	-2	-4	16	-1	1	25	22	S2	930	0	23C1
164	TABDMP3*	5	DIT	-2	-4	16	-1	1	35	23	S2	930	0	23C2
165	TABDMP4*	5	DIT	-2	-4	16	-1	1	45	24	S2	960	0	23C3
166	FREQ2	7	DYNAMICS	-2	4	8	705	0	1107	11	S1	1660	0	23C4
167	CQUAD3	8	GEØM2	2	8	12	325	1	2958	40	S1	3460	-1	23C5
168	PQUAD3	2	EPT	2	-4	8	-1	1	852	49	S1	1690	-1	23C6
169	SPQUAD3	2	EPT	2	-4	8	-1	1	6702	67	S1	1690	-1	23D1
170	SETR	8	GEØM2	2	-4	9	-1	0	7608	76	S2	1060	-1	23D2
171	SPTRIA1	2	EPT	2	4	16	221	1	7402	74	S1	2980	-1	23D3
172	SPTRIA2	2	EPT	2	4	8	237	0	7502	75	S1	3000	-1	23D4
173	SPTRMEM	2	EPT	2	4	8	237	0	7702	77	S1	3000	-1	23D5

Table 1(f). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
174	SPTRBSC	2	EPT	2	4	12	257	1	7302	73	S1	3020	-1	23D6
175	SPTRPLT	2	EPT	2	4	12	257	1	7802	78	S1	3020	-1	23E1
176	SECL	8	GEØM2	2	-4	9	-1	0	7308	73	S2	1060	-1	23E2
177	SECP	8	GEØM2	2	-4	9	-1	0	7408	74	S2	1060	-1	23E3
178	SPTRIA3	2	EPT	2	-4	8	-1	1	7602	76	S1	1690	-1	23E4
179	BARØR	8	GEØM2	0	4	12	25	2	0	0	S1	100	-1	23E5
180	CBAR	8	GEØM2	0	8	20	73	1	2408	24	S1	200	-1	23E6
181	PBAR	2	EPT	0	4	24	621	1	52	20	S1	300	-1	31A1
182	DAREA	7	DYNAMICS	-2	4	8	101	0	27	17	S3	1820	0	31A2
183	DELAY	7	DYNAMICS	-2	4	8	101	0	37	18	S3	1820	0	31A3
184	DPHASE	7	DYNAMICS	-2	4	8	101	0	77	19	S3	1820	0	31A4
185	PLFACT	3	MPT	-2	4	8	-1	1	1103	11	S3	1430	-1	31A5
186	CGENEL	8	GEØM2	2	-4	8	-1	1	2008	20	S3	1860	-1	31A6
187	PGENEL	2	EPT	2	4	8	573	0	352	26	S1	1870	-1	31B1
188	ELDELE	13	FØRCE	2	4	8	37	0	1913	19	S3	1600	-1	31B2
189	MATT3	3	MPT	-2	4	16	525	1	1503	15	S4	1020	-1	31B3
190	RFØRCE	9	GEØM3	-2	8	12	109	0	5509	55	S1	1900	-1	31B4
191	TABRND1	5	DIT	-2	-4	16	-1	1	55	25	S2	930	0	31B5
192	TABRND2*	5	DIT	-2	-4	16	-1	1	65	26	S2	930	0	31B6
193	TABRND3*	5	DIT	-2	-4	16	-1	1	75	27	S2	930	0	31C1
194	TABRND4*	5	DIT	-2	-4	16	-1	1	85	28	S2	960	0	31C2
195	RANDPS	7	DYNAMICS	-2	4	12	782	0	2107	21	S4	1950	-1	31C3
196	RANDT1	7	DYNAMICS	-2	4	8	752	0	2207	22	S4	1960	-1	31C4
197	RANDT2	7	DYNAMICS	-2	-4	8	-1	1	2307	23	XX	XXXX	-1	31C5
198	PLØAD1	9	GEØM3	-2	-	-	-	-	6909	69	XX	XXXX	-1	31C6
199	PLØAD2	9	GEØM3	-2	4	9	774	0	6802	68	S4	1990	-1	31D1
200	DTI	12	PØØL	0	-4	16	-1	0	0	0	S2	2000	0	31D2

Table 2(a). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

The following list gives an explanation of the column headings on the following pages of Table 2.

A = Internal IFP Bulk Data Card Number

B = Bulk Data Card Name (an asterisk following a name implies the card is not available)

C = Internal IFP GINØ Output File Number

D = Data Block Name

E = Approach Acceptance Indicator

- -2 = Illegal for the Force Approach
- -1 = Not Used by the Force Approach
- 0 = OK for any Approach
- 1 = Not Used by the Displacement Approach
- 2 = Illegal for the Displacement Approach

F = Minimum Number of Words Allowed Per Logical Card (F negative implies an open ended card)

G = Maximum Number of Words Allowed Per Logical Card

H = Format Check Pointer Into IFX7BD

I = Field 2 Uniqueness Check Flag

- 0 = No Check is Made
- 1 = Check is Made
- 2 = Special

J = Subroutine LØCATE Code for Card on Output Data Block

K = Trailer Bit Position

L = Pointer to Secondary (Card Dependent) Code

- S1 = Subroutine IFS1P
- S2 = Subroutine IFS2P
- S3 = Subroutine IFS3P
- S4 = Subroutine IFS4P

M = FØRTRAN Statement Number in the Card Dependent Subroutines

N = Conical Shell Problem Flag

- -1 = Illegal for Shell Mode

Table 2(b). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

0 = OK for Shell Mode

1 = Puts Card Into Different Data Block

O = Users Map for Data Blocks IFX2BD,...,IFX6BD

Values for I = 1,2 or 3

J = 1,2 or 3

H = A,B,C,D or E

K = 1,2,3,4,5 or 6

I = Is Data Statement in the Block Data Program

J = The Group of A Through E Continuation Card Blocks Within the Ith Data Statement

H = Alphabetic Character in Col 6 (Continuation Column) in the Jth Group

K = The Pair Number on Line H Where the Actual Data is

Table 2(c). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
89		1		0	8	8	-1	2			S2	890	0	13E5
144	AXIC	15	AXIC	-2	4	8	93	0	515	5	S3	1440	0	22D6
179	BARØR	8	GEØM2	0	4	12	25	2	0	0	S1	100	-1	23E5
3	BEAMOR*	8	GEØM2	-2	4	12	25	2	0	0	S3	300	-1	11A3
180	CBAR	8	GEØM2	0	8	20	73	1	2408	24	S1	200	-1	23E6
51	CBEAM*	8	GEØM2	-2	8	24	289	1	101	1	S3	3340	-1	12D3
146	CCØNEAX	15	AXIC	-2	4	8	645	1	2315	23	S3	1460	0	22E2
69	CDAMP1	8	GEØM2	-2	4	12	337	1	201	2	S1	3620	-1	13B3
70	CDAMP2	8	GEØM2	-2	4	12	397	1	301	3	S1	3623	-1	13B4
71	CDAMP3	8	GEØM2	-2	4	8	37	0	401	4	S1	3675	-1	13B5
72	CDAMP4	8	GEØM2	-2	4	8	409	0	501	5	S1	3698	-1	13B6
73	CELAS1	8	GEØM2	-2	4	12	337	1	601	6	S1	3620	-1	13C1
74	CELAS2	8	GEØM2	-2	4	12	417	1	701	7	S1	3800	-1	13C2
75	CELAS3	8	GEØM2	-2	4	8	37	0	801	8	S1	3676	-1	13C3
76	CELAS4	8	GEØM2	-2	4	8	409	0	901	9	S1	3699	-1	13C4
186	CGENEL	8	GEØM2	2	-4	8	-1	1	2008	20	S3	1860	-1	31A6
65	CMASS1	8	GEØM2	-2	4	12	337	1	1001	10	S1	3620	-1	13A5
66	CMASS2	8	GEØM2	-2	4	12	397	1	1101	11	S1	3623	-1	13A6
67	CMASS3	8	GEØM2	-2	4	8	37	0	1201	12	S1	3674	-1	13B1
68	CMASS4	8	GEØM2	-2	4	8	409	0	1301	13	S1	3697	-1	13B2
63	CØNM1	8	GEØM2	0	8	28	349	1	1401	14	S1	3580	-1	13A3
64	CØNM2	8	GEØM2	0	8	20	377	1	1501	15	S1	3600	-1	13A4
47	CØNRØD	8	GEØM2	0	8	12	277	1	1601	16	S1	3260	-1	12C5
6	CØRD1C	1	GEØM1	0	4	8	37	0	1701	17	S1	600	-1	11A6
5	CØRD1R	1	GEØM1	0	4	8	37	0	1801	18	S1	500	-1	11A5
7	CØRD1S	1	GEØM1	0	4	8	37	0	1901	19	S1	700	-1	11B1
9	CØRD2C	1	GEØM1	0	12	16	45	1	2001	20	S1	900	-1	11B3
8	CØRD2R	1	GEØM1	0	12	16	45	1	2101	21	S1	800	-1	11B2
10	CØRD2S	1	GEØM1	0	12	16	45	1	2201	22	S1	1000	-1	11B4
60	CQDMEM	8	GEØM2	0	8	12	325	1	2601	26	S1	3460	-1	12E6
59	CQDPLT	8	GEØM2	0	8	12	325	1	2701	27	S1	3460	-1	12E5
57	CQUAD1	8	GEØM2	0	8	12	325	1	2801	28	S1	3460	-1	12E3
58	CQUAD2	8	GEØM2	0	8	12	325	1	2901	29	S1	3460	-1	12E4
167	CQUAD3	8	GEØM2	2	8	12	325	1	2958	40	S1	3460	-1	23C5
48	CRØD	8	GEØM2	0	4	8	37	0	3001	30	S1	3281	-1	12C6
61	CSHEAR	8	GEØM2	0	8	12	337	1	3101	31	S1	3540	-1	13A1
104	CTØRDRG	8	GEØM2	-2	4	12	750	1	1908	19	S4	1040	-1	21C2
80	CTRAPRG	8	GEØM2	-2	8	12	737	1	1808	18	S4	800	-1	13D2
54	CTRBSC	8	GEØM2	0	8	12	313	1	3201	32	S1	3360	-1	12D6
52	CTRIA1	8	GEØM2	0	8	12	313	1	3301	33	S1	3360	-1	12D4
53	CTRIA2	8	GEØM2	0	8	12	313	1	3401	34	S1	3360	-1	12D5
114	CTRIA3	8	GEØM2	2	8	12	313	1	3458	41	S1	3360	-1	21D6
79	CTRIARG	8	GEØM2	-2	8	12	738	1	1708	17	S4	790	-1	13D1
56	CTRMEM	8	GEØM2	0	8	12	313	1	3501	35	S1	3360	-1	12E2
55	CTRPLT	8	GEØM2	0	8	12	313	1	3601	36	S1	3360	-1	12E1
49	CTUBE	8	GEØM2	0	4	8	37	0	3701	37	S1	3282	-1	12D1
62	CTWIST	8	GEØM2	0	8	12	337	1	3801	38	S1	3540	-1	13A2
50	CVISC	8	GEØM2	-2	4	8	37	0	3901	39	S1	3283	-1	12D2
182	DAREA	7	DYNAMICS	-2	4	8	101	0	27	17	S3	1820	0	31A2
81	DEFØRM	4	EDT	-2	4	8	157	0	104	1	S1	2500	-1	13D3
183	DELAY	7	DYNAMICS	-2	4	8	101	0	37	18	S3	1820	0	31A3
161	DFØRM	4	EDT	2	4	8	101	0	204	2	S3	1610	-1	23B5
123	DLØAD	7	DYNAMICS	-2	4	8	-1	1	57	5	S3	4060	0	22A3
119	DMI	12	PØØL	0	-4	16	-1	0	0	0	S2	1190	0	21E5
120	DMIG	14	MATPØØL	-2	-4	12	-1	0	114	1	S2	1200	0	21E6
184	DPHASE	7	DYNAMICS	-2	4	8	101	0	77	19	S3	1820	0	31A4
143	DSFACT	3	MPT	-2	4	8	-1	1	53	10	S3	1430	0	22D5

Table 2(d). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
200	DTI	12	PØØL	0	-4	16	-1	0	0	0	S2	2000	0	31D2
160	EDFIR	13	FØRCE	2	4	8	37	0	1513	15	S3	1600	-1	23B4
86	EIGB	7	DYNAMICS	-2	14	18	469	1	107	11	S2	850	0	13E2
87	EIGC	7	DYNAMICS	-2	-4	10	-1	1	207	2	S2	870	0	13E3
158	EIGP	7	DYNAMICS	-2	4	8	561	0	257	4	S1	1580	0	23B2
85	EIGR	7	DYNAMICS	-2	14	18	469	1	307	3	S2	850	0	13E1
188	ELDELE	13	FØRCE	2	4	8	37	0	1913	19	S3	1600	-1	31B2
124	EPØINT	7	DYNAMICS	-2	4	9	794	0	707	7	S4	1050	0	22A4
18	FØRCE	9	GEØM3	0	8	12	109	0	4201	42	S1	1800	1	11C6
20	FØRCE1	9	GEØM3	0	8	12	121	0	4001	40	S1	2000	-1	11D2
22	FØRCE2	9	GEØM3	0	8	12	133	0	4101	41	S1	2200	-1	11D4
156	FØRCEAX	15	AXIC	-2	4	12	109	0	2115	21	S3	1560	0	23A6
126	FREQ	7	DYNAMICS	-2	4	8	-1	1	1307	13	S3	1260	0	22A6
125	FREQ1	7	DYNAMICS	-2	4	8	705	0	1007	10	S1	1250	0	22A5
166	FREQ2	7	DYNAMICS	-2	4	8	705	0	1107	11	S1	1660	0	23C4
28	GENEL	8	GEØM2	-2	-4	9	-1	1	4301	43	S3	2800	-1	11E4
26	GRAV	9	GEØM3	0	8	12	165	1	4401	44	S1	2600	1	11E2
2	GRDSET	1	GEØM1	0	4	12	13	2	0	0	S3	200	-1	11A2
1	GRID	1	GEØM1	0	4	12	1	1	4501	45	S3	100	-1	11A1
84	LØAD	9	GEØM3	0	4	8	-1	1	4551	61	S3	4060	1	13D6
159	MASSC	13	FØRCE	2	4	8	-1	0	3313	33	S3	1590	-1	23B3
77	MAT1	3	MPT	0	4	20	429	1	103	1	S1	3860	0	13C5
78	MAT2	3	MPT	0	8	20	449	1	203	2	S1	3880	-1	13C6
122	MAT3	3	MPT	-2	4	20	449	1	1403	14	S4	1220	-1	22A2
90	MATS1	3	MPT	-2	4	16	545	1	503	5	S4	900	-1	13E6
103	MATS2*	3	MPT	-2	4	16	525	1	603	6	S4	1020	-1	21C1
91	MATT1	3	MPT	0	4	16	545	1	703	7	S4	900	0	21A1
102	MATT2	3	MPT	0	4	16	525	1	803	8	S4	1020	-1	21B6
189	MATT3	3	MPT	-2	4	16	525	1	1503	15	S4	1020	-1	31B3
157	MØMAX	15	AXIC	-2	4	12	109	0	3815	38	S3	1560	0	23B1
19	MØMENT	9	GEØM3	0	8	12	109	0	4801	48	S1	1800	1	11D1
21	MØMENT1	9	GEØM3	0	8	12	121	0	4601	46	S1	2000	-1	11D3
23	MØMENT2	9	GEØM3	0	8	12	133	0	4701	47	S1	2200	-1	11D5
17	MPC	10	GEØM4	-2	4	8	-1	0	4901	49	S3	1700	-1	11C5
83	MPCADD	10	GEØM4	-2	4	8	-1	1	4891	60	S3	4020	1	13D5
149	MPCAX	15	AXIC	-2	4	8	-1	0	4015	40	S3	1490	0	22E5
127	NØLIN1	7	DYNAMICS	-2	8	12	725	0	3107	31	S1	1270	0	22B1
128	NØLIN2	7	DYNAMICS	-2	8	12	725	0	3207	32	S1	1280	0	22B2
129	NØLIN3	7	DYNAMICS	-2	8	12	725	0	3307	33	S1	1290	0	22B3
130	NØLIN4	7	DYNAMICS	-2	8	12	725	0	3407	34	S1	1290	0	22B4
15	ØMIT	10	GEØM4	-2	4	8	37	0	5001	50	S1	1400	-1	11C3
92	ØMIT	10	GEØM4	-2	-4	8	-1	0	4951	63	S3	3981	-1	21A2
150	ØMITAX	15	AXIC	-2	4	8	337	0	4315	43	S3	1500	0	22E6
82	PARAM	6	PVT	0	-5	16	-1	2	0	0	S3	3960	0	13D4
181	PBAR	2	EPT	0	4	24	621	1	52	20	S1	300	-1	31A1
32	PBEAM*	2	EPT	0	8	24	197	1	102	1	S1	2960	-1	12A2
147	PCØNEAX	2	EPT	-2	4	28	653	1	152	19	S3	1470	0	22E3
45	PDAMP	2	EPT	-2	4	8	269	0	202	2	S1	3200	-1	12C3
46	PELAS	2	EPT	-2	4	8	497	0	302	3	S1	3240	-1	12C4
187	PGENEL	2	EPT	2	4	8	573	0	352	26	S1	1870	-1	31B1
185	PLFACT	3	MPT	-2	4	8	-1	1	1103	11	S3	1430	-1	31A5
24	PLØAD	9	GEØM3	0	8	12	145	0	5101	51	S1	2400	-1	11D6
198	PLØAD1	9	GEØM3	-2	-	-	-	-	6909	69	XX	XXXX	-1	31C6
199	PLØAD2	9	GEØM3	-2	4	9	774	0	6802	68	S4	1990	-1	31D1
11	PLØTEL	8	GEØM2	0	4	8	505	0	5201	52	S1	1111	-1	11B5
44	PMASS	2	EPT	-2	4	8	269	0	402	4	S1	3200	-1	12C2

Table 2(e). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
152	PØINTAX	15	AXIC	-2	4	8	517	1	4915	49	S3	1520	0	23A2
41	PQDMEM	2	EPT	0	4	8	237	0	502	5	S1	3000	-1	12B5
40	PQDPLT	2	EPT	0	4	12	257	0	602	6	S1	3020	-1	12B4
38	PQUAD1	2	EPT	0	4	16	221	0	702	7	S1	2980	-1	12B2
39	PQUAD2	2	EPT	0	4	8	237	0	802	8	S1	3000	-1	12B3
168	PQUAD3	2	EPT	2	-4	8	-1	1	852	49	S1	1690	-1	23C6
154	PRESAX	15	AXIC	-2	4	12	61	0	5215	52	S3	1540	0	23A4
29	PRØD	2	EPT	0	4	12	165	1	902	9	S1	2900	-1	11E5
42	PSHEAR	2	EPT	0	4	8	237	0	1002	10	S1	3000	-1	12B6
121	PTØRDRG	2	EPT	-2	4	8	237	0	2102	21	S1	3000	-1	22A1
35	PTRBSC	2	EPT	0	4	12	257	1	1102	11	S1	3020	-1	12A5
33	PTRIA1	2	EPT	0	4	16	221	1	1202	12	S1	2980	-1	12A3
34	PTRIA2	2	EPT	0	4	8	237	0	1302	13	S1	3000	-1	12A4
115	PTRIA3	2	EPT	2	-4	8	-1	1	1352	24	S1	1690	-1	21E1
37	PTRMEM	2	EPT	0	4	8	237	0	1402	14	S1	3000	-1	12B1
36	PTRPLT	2	EPT	0	4	12	257	1	1502	15	S1	3020	-1	12A6
30	PTUBE	2	EPT	0	4	12	177	1	1602	16	S1	2920	-1	11E6
43	PTWIST	2	EPT	0	4	8	237	0	1702	17	S1	3000	-1	12C1
31	PVISC	2	EPT	-2	4	8	189	0	1802	18	S1	310	-1	12A1
108	QDSEP	8	GEØM2	2	4	8	37	0	4808	48	S4	1080	-1	21C6
195	RANDPS	7	DYNAMICS	-2	4	12	782	0	2107	21	S4	1950	-1	31C3
196	RANDT1	7	DYNAMICS	-2	4	8	752	0	2207	22	S4	1960	-1	31C4
197	RANDT2	7	DYANMICS	-2	-4	8	-1	1	2307	23	XX	XXXX	-1	31C5
88	REACT	8	GEØM2	2	8	24	597	1	5251	62	S4	880	-1	13E4
190	RFØRCE	9	GEØM3	-2	8	12	109	0	5509	55	S1	1900	1	31B4
145	RINGAX	15	AXIC	-2	4	12	245	1	5615	56	S3	1450	0	22E1
131	RLØAD1	7	DYNAMICS	-2	8	8	337	1	5107	51	S3	1310	0	22B5
132	RLØAD2	7	DYNAMICS	-2	8	8	337	1	5207	52	S3	1310	0	22B6
176	SECL	8	GEØM2	2	-4	9	-1	0	7308	73	S2	1060	-1	23E2
177	SECP	8	GEØM2	2	-4	9	-1	0	7408	74	S2	1060	-1	23E3
153	SECTAX	15	AXIC	-2	4	12	177	1	6015	60	S3	1530	0	23A3
106	SEQD	8	GEØM2	2	-4	9	-1	0	7508	75	S2	1060	-1	21C4
107	SEQDBFE	8	GEØM2	2	4	8	37	0	4608	46	S2	1070	-1	21C5
135	SEQEP	7	DYNAMICS	-1	4	8	37	0	5707	57	S1	40	-1	22C3
4	SEQGP	1	GEØM1	-1	4	8	37	0	5301	53	S1	40	-1	11A4
170	SETR	8	GEØM2	2	-4	9	-1	0	7608	76	S2	1060	-1	23D2
116	SETRBFE	8	GEØM2	2	4	8	549	0	4708	47	S1	1160	-1	21E2
25	SLØAD	9	GEØM3	-2	4	8	157	0	5401	54	S1	2500	-1	11E1
16	SPC	10	GEØM4	-2	4	8	101	0	5501	55	S1	1600	-1	11C4
12	SPC1	10	GEØM4	-2	4	8	-1	0	5481	58	S3	3980	-1	11B6
13	SPCADD	10	GEØM4	-2	4	8	-1	1	5491	59	S3	4020	1	11C1
148	SPCAX	15	AXIC	-2	4	12	485	0	6215	62	S3	1480	0	22E4
105	SPØINT	8	GEØM2	-2	4	9	794	0	5551	49	S4	1050	-1	21C3
111	SPQDMEM	2	EPT	2	4	8	237	0	6302	63	S1	3000	-1	21D3
112	SPQDPLT	2	EPT	2	4	12	257	0	6402	64	S1	3020	-1	21D4
109	SPQUAD1	2	EPT	2	4	16	221	0	6502	65	S1	2980	-1	21D1
110	SPQUAD2	2	EPT	2	4	8	237	0	6602	66	S1	3000	-1	21D2
169	SPQUAD3	2	EPT	2	-4	8	-1	1	6702	67	S1	1690	-1	23D1
174	SPTRBSC	2	EPT	2	4	12	257	1	7302	73	S1	3020	-1	23D6
171	SPTRIA1	2	EPT	2	4	16	221	1	7402	74	S1	2980	-1	23D3
172	SPTRIA2	2	EPT	2	4	8	237	0	7502	75	S1	3000	-1	23D4
178	SPTRIA3	2	EPT	2	-4	8	-1	1	7602	76	S1	1690	-1	23E4
173	SPTRMEM	2	EPT	2	4	8	237	0	7702	77	S1	3000	-1	23D5
175	SPTRPLT	2	EPT	2	4	12	257	1	7802	78	S1	3020	-1	23E1
151	SUPAX	15	AXIC	-2	4	8	337	0	6415	64	S3	1500	0	23A1
14	SUPØRT	10	GEØM4	-2	4	8	37	0	5601	56	S1	1400	-1	11C2
162	TABDMP1	5	DIT	-2	-4	16	-1	1	15	21	S2	930	0	23B6
163	TABDMP2*	5	DIT	-2	-4	16	-1	1	25	22	S2	930	0	23C1

Table 2(f). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
164	TABDMP3*	5	DIT	-2	-4	16	-1	1	35	23	S2	930	0	23C2
165	TABDMP4*	5	DIT	-2	-4	16	-1	1	45	24	S2	960	0	23C3
133	TABLED1	5	DIT	-2	-4	16	-1	1	1105	11	S2	930	0	22C1
134	TABLED2	5	DIT	-2	-4	16	-1	1	1205	12	S2	930	0	22C2
140	TABLED3	5	DIT	-2	-4	16	-1	1	1305	13	S2	930	0	22D2
141	TABLED4	5	DIT	-2	-4	16	-1	1	1405	14	S2	960	0	22D3
93	TABLEM1	5	DIT	0	-4	16	-1	1	105	1	S2	930	0	21A3
94	TABLEM2	5	DIT	0	-4	16	-1	1	205	2	S2	930	0	21A4
95	TABLEM3	5	DIT	0	-4	16	-1	1	305	3	S2	930	0	21A5
96	TABLEM4	5	DIT	0	-4	16	-1	1	405	4	S2	960	0	21A6
97	TABLES1	5	DIT	-2	-4	16	-1	1	3105	31	S2	930	-1	21B1
99	TABLES2*	5	DIT	-2	-4	16	-1	1	3205	32	S2	930	-1	21B3
100	TABLES3*	5	DIT	-2	-4	16	-1	1	3305	33	S2	930	-1	21B4
101	TABLES4*	5	DIT	-2	-4	16	-1	1	3405	34	S2	960	-1	21B5
191	TABRND1	5	DIT	-2	-4	16	-1	1	55	25	S2	930	0	31B5
192	TABRND2*	5	DIT	-2	-4	16	-1	1	65	26	S2	930	0	31B6
193	TABRND3*	5	DIT	-2	-4	16	-1	1	75	27	S2	930	0	31C1
194	TABRND4*	5	DIT	-2	-4	16	-1	1	85	28	S2	960	0	31C2
27	TEMP	9	GEØM3	0	4	8	157	0	5701	57	S1	2500	-1	11E3
155	TEMPAX	15	AXIC	-2	4	8	237	0	6815	68	S3	1550	0	23A6
98	TEMPD	9	GEØM3	0	4	12	269	0	5641	65	S4	980	-1	21B2
136	TF	7	DYNAMICS	-2	8	12	-1	0	6207	62	S1	1360	0	22C4
137	TIC	7	DYNAMICS	-2	4	12	713	0	6607	66	S1	1370	0	22C5
138	TLØAD1	7	DYNAMICS	-2	8	8	681	1	7107	71	S3	1380	0	22C6
139	TLØAD2	7D	DYNAMICS	-2	8	16	689	1	7207	72	S3	1390	0	22D1
142	TSTEP	7	DYNAMICS	-2	4	8	-1	1	8307	83	S1	1420	0	22D4
117	VECDN	13	FØRCE	2	8	12	177	1	8313	83	S4	1170	-1	21E3
118	VECGP	13	FØRCE	2	8	12	585	1	8413	84	S4	1180	-1	21E4
113	ZI	13	FØRCE	2	8	12	485	0	8913	89	S4	1130	-1	21D5

4.6 EXECUTIVE PREFACE MODULE IFP3 (INPUT FILE PROCESSOR 3)

4.6.1 Entry Point: IFP3

4.6.2 Purpose

1. To interpret Bulk Data cards unique to an axisymmetric conical shell problem.

2. To generate and distribute to data blocks GEØM1, GEØM2, GEØM3, and GEØM4 data cards for all harmonics specified in the problem.

3. To convert the following input Bulk Data cards to the following output Bulk Data cards.

Input Bulk Data Card Type	Input Data Block	Output Bulk Data Card Type	Output Data Block
AXIC	AXIC	None	
CCØNEAX	AXIC	CCØNE	GEØM2
FØRCEAX	AXIC	FØRCE	GEØM3
FØRCE	AXIC	FØRCE	GEØM3
GRAV	AXIC	GRAV	GEØM3
LØAD	AXIC	LØAD	GEØM3
MØMAX	AXIC	MØMENT	GEØM3
MØMENT	AXIC	MØMENT	GEØM3
MPCADD	AXIC	MPCADD	GEØM4
MPCAX	AXIC	MPC	GEØM4
ØMITAX	AXIC	ØMIT	GEØM4
PØINTAX	AXIC	MPC GRID	GEØM4 GEØM1
PRESAX	AXIC	PRESAX	GEØM3
RINGAX	AXIC	SPC GRID	GEØM4 GEØM1
SECTAX	AXIC	MPC GRID	GEØM4 GEØM1
SEQGP	AXIC	SEQGP	GEØM1
SPCADD	AXIC	SPCADD	GEØM4
SPCAX	AXIC	SPC	GEØM4
SUPAX	AXIC	SUPØRT	GEØM4
TEMPAX	AXIC	TEMP	GEØM3
TEMPD	AXIC	TEMPD	GEØM3

4.6.3 Calling Sequence

CALL IFP3. IFP3, a Preface module, is called only from the Preface driver SEMINT.

4.6.4 Input Data Blocks

AXIC - Bulk Data Deck cards as output from IFP.

4.6.5 Output Data Blocks

GEØM1 - Grid Point Data.

GEØM2 - Element Connection Data.

GEØM3 - Loading Data.

GEØM4 - Constraint Data.

4.6.6 Parameters

Not applicable to IFP3.

4.6.7 Method

In the following a "card" is defined as the type of card image output by IFP, i.e., with the mneumonic stripped off and a card image, in some cases a modified card image, written on the output data block.

4.6.7.1 Initialization and Overall Method

IFP3 first determines the amount of core available, and then allocates three buffers for the SCRATCH, AXIC, and GEØM data blocks. The AXIC data block is opened for input using PRELØC. At this point the 21 types of input Bulk Data cards are found, one type at a time, on the AXIC data block, using the LØCATE routine. The cards of each type are converted and output to a GEØM data block.

To facilitate the operation, all Bulk Data cards causing output to a particular GEØM data block are handled together. Thus the processing is such that all cards affecting GEØM2 are first converted, and following these all those for GEØM3, then GEØM4, and finally GEØM1.

The superscripts s and c in the following refer to the sine and cosine sets respectively.

4.6.7.2 Conversion of Input Bulk Data Cards to Output Bulk Data Cards for GEØM2.

1. AXIC card

-Input Card-

AXIC	Harm	0

-Output Card-
(None)

The AXIC card supplies Harm which is used to compute the number of harmonics, N.

$$N = Harm + 1 \tag{1}$$

2. CCØNEAX card

-Input Card-

CCØNEAX	EL-ID	Prop-ID	$Ring_A$ ID	$Ring_B$ ID

-N Output Cards-

$CCØNE_n$	$EL\text{-}ID_n$	Prop-ID	$Ring_A$ ID_n	$Ring_B$ ID_n

where

$$EL\text{-}ID_n = EL\text{-}ID \times 1000 + n, \tag{2}$$

$$Ring_A\ ID_n = Ring_A\ ID + 1000000 \times n, \tag{3}$$

$$Ring_B\ ID_n = Ring_B\ ID + 1000000 \times n, \tag{4}$$

for n = 1, 2, ..., N.

4.6.7.3 Conversion of Input Bulk Data Cards to Output Cards for GEØM3.

1. FØRCE card and MØMENT card

These two cards are output to GEØM3 as input from AXIC.

2. FØRCEAX card and MØMAX card

-Input Cards-

FORCEAX	Set ID	Ring ID	Harm ID	Factor	F_r	F_ϕ	F_z
MØMAX	Set ID	Ring ID	Harm ID	Factor	M_r	M_ϕ	M_z

-Output Cards-

FØRCE	Set ID	Ring ID_H	0	Factor	F_r	F_ϕ	F_z
MØMENT	Set ID	Ring ID_H	0	Factor	M_r	M_ϕ	M_z

where

$$\text{Ring ID}_H = \text{Ring ID} + (\text{Harm ID} + 1) \times 1000000 \qquad (5)$$

If FØRCE cards and FØRCEAX cards both exist, then the resulting output cards of the FØRCE and FØRCEAX cards are merged in sort on Set ID's. This applies to MØMENT and MØMAX cards also.

3. GRAV card, LØAD card, and TEMPD card

These three cards are output to GEØM3 as they are input from AXIC.

4. PRESAX card

-Input Card-

PRESAX	Set ID	Value	Ring_A ID	Ring_B ID	ϕ_1	ϕ_2

-N Output Cards-

$PRESAX_n$	Set ID	Value	$Ring_A ID_n$	$Ring_B ID_n$	ϕ_1	ϕ_2	n

where

$$Ring_A\ ID_n = Ring_A\ ID + 1000000 \times n \tag{6}$$

$$Ring_B\ ID_n = Ring_B\ ID + 1000000 \times n \tag{7}$$

for n = 1, 2, ..., N.

5. TEMPAX card

-Input Card-

TEMPAX	Set ID	Ring ID	ϕ_1	T_1	ϕ_2	T_2	ϕ_3	T_3

-N x 2 Output Cards -

$N\text{-}TEMP^S_{n+1}$	Set ID^S	$Ring\ ID_{n+1}$	T^S_n

$N\text{-}TEMP^C_{n+1}$	Set ID^C	$Ring\ ID_{n+1}$	T^C_n

where

$$Ring\ ID_{n+1} = Ring\ ID + 1000000 \times (n+1) \tag{8}$$

$$Set\ ID^S = Set\ ID + 100000000 \tag{9}$$

$$Set\ ID^C = Set\ ID + 200000000 \tag{10}$$

for n = 0, 1, ..., N-1.

T^S_n and T^C_n are computed as follows. All TEMPAX cards having the same Set ID and Ring ID are gathered together. The angles ϕ_i and temperatures T_i are arrayed into a two-

dimensional matrix.

ϕ_1	T_1
ϕ_2	T_2
⋮	⋮
ϕ_k	T_k

where k is the total number of unique ϕ's, and the ϕ's are converted to radians.

The matrix is sorted on the ϕ column, and duplicate angles are removed.

For n = 0, we have

$$T_0^C = \frac{1}{4\pi} \sum_{i=1}^{k} (T_i + T_{i+1}) (\phi_{i+1} - \phi_i), \tag{11}$$

and

$$T_0^S = 0. \tag{12}$$

For n > 0

$$T_n^C = \frac{1}{\pi} \sum_{i=1}^{k} \frac{1}{(\phi_{i+1} - \phi_i)} \left\{ \frac{(T_i\phi_{i+1} - T_{i+1}\phi_i)}{n} (\sin n\phi_{i+1} - \sin n\phi_i) \right.$$

$$\left. + \frac{T_{i+1} - T_i}{n^2} (\cos n\phi_{i+1} - \cos n\phi_i + n\phi_{i+1} \sin n\phi_{i+1} - n\phi_i \sin n\phi_i) \right\} \tag{13}$$

and

$$T_n^S = \frac{1}{\pi} \sum_{i=1}^{k} \frac{1}{(\phi_{i+1} - \phi_1)} \left\{ \frac{(T_i\phi_{i+1} - T_{i+1}\phi_i)}{n} (-\cos n\phi_{i+1} + \cos n\phi_i) \right.$$

$$\left. + \frac{T_{i+1} - T_i}{n^2} (\sin n\phi_{i+1} - \sin n\phi_i - n\phi_{i+1} \cos n\phi_{i+1} + n\phi_i \cos n\phi_i) \right\} \tag{14}$$

where

$$\phi_{k+1} = \phi_1 + 2\pi , \tag{15}$$

and

$$T_{k+1} = T_1 . \tag{16}$$

4.6.7.4 Conversion of Input Bulk Data Cards to Output Cards for GEØM4

1. SPCADD card

-Input Card-

SPCADD	Set ID	S1	...	S9	-1	} Open-ended

-2 Output Cards-

$SPCADD^S$	$Set\ ID^S$	S1	...	S9	101	-1
$SPCADD^C$	$Set\ ID^C$	S1	...	S9	102	-1

where

$$Set\ ID^S = Set\ ID + 100000000\ , \tag{17}$$

$$Set\ ID^C = Set\ ID + 200000000\ . \tag{18}$$

The following 4 mandatory SPCADD cards are also created and put out on GEØM4 regardless of whether there are or are not SPCADD cards present.

SPCADD	100000101	101	-1
SPCADD	200000102	102	-1
SPCADD	100000000	101	-1
SPCADD	200000000	102	-1

2. MPCADD card

All operations performed for the SPCADD card are performed for the MPCADD card.

3. SPCAX card

-Input Card-

SPCAX	Set ID	Ring ID	Harm ID	Comp	Value

-3 Output Cards-

$SPCADD^S$	$Set\ ID^S$	Set ID	101	-1

$SPCADD^C$	$Set\ ID^C$	Set ID	102	-1

where

$$Set\ ID^S = Set\ ID + 100000000\ , \tag{19}$$

$$Set\ ID^C = Set\ ID + 200000000\ ; \tag{20}$$

and

SPC	Set ID	$Ring\ ID_H$	Comp	Value

where

$$Ring\ ID_H = Ring\ ID + (Harm\ ID + 1) \times 1000000. \tag{21}$$

4. MPCAX card

-Input Card-

MPCAX	Set ID	Ring ID	Harm ID	Comp ID	Value	...	-1	-1	-1	-1	Open Ended

Repeats

-3 Output Cards-

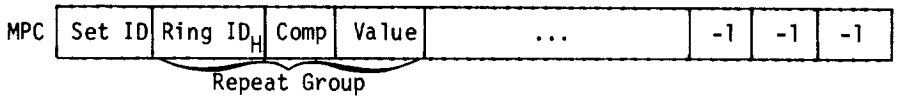
$MPCADD^S$	$Set\ ID^S$	Set ID	101	-1

$MPCADD^C$	$Set\ ID^C$	Set ID	102	-1

$$Set\ ID^S = Set\ ID + 100000000\ , \tag{22}$$

$$Set\ ID^C = Set\ ID + 200000000\ . \tag{23}$$

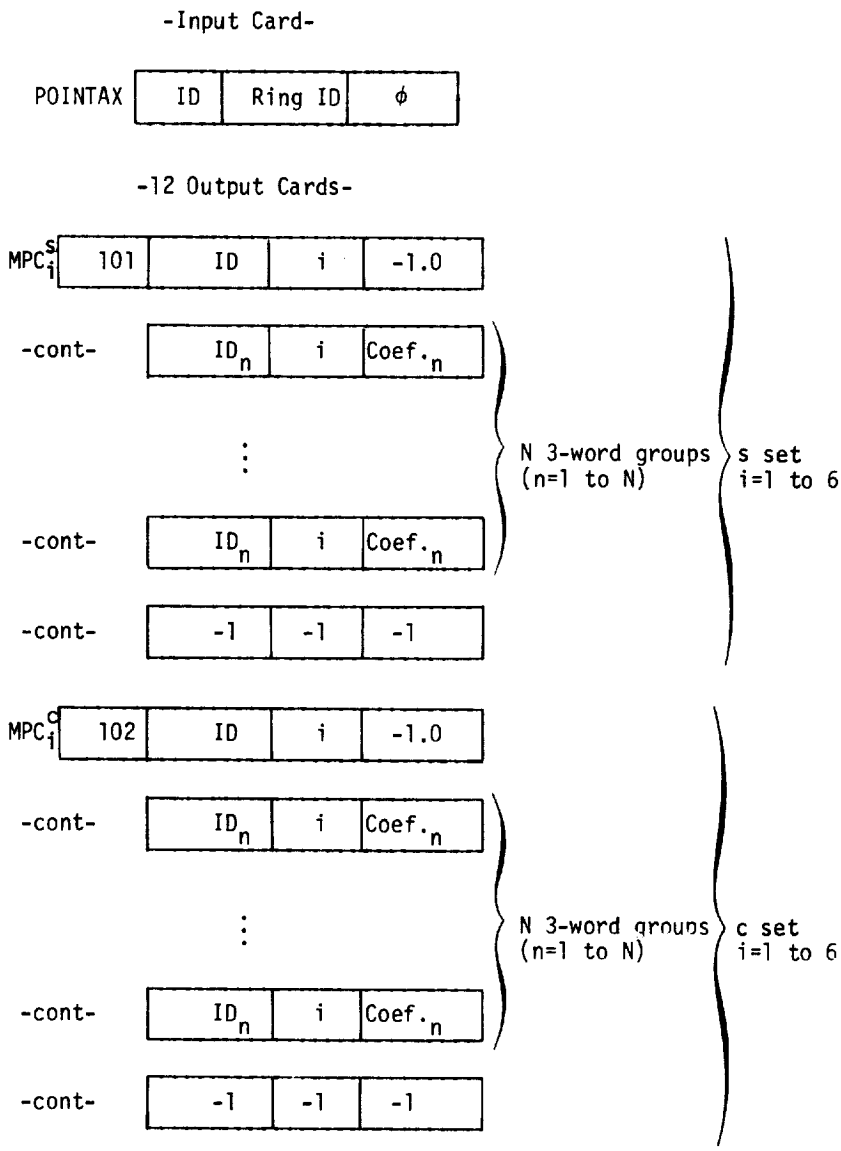
and



For each 3-word group output,

$$\text{Ring ID}_H = \text{Ring ID} + (\text{Harm ID} + 1) \times 1000000. \quad (24)$$

5. PØINTAX card (also processed in GEØM1 section)



where the $Coef._n$ above are defined by:

i	c set-$Coef._n$	s set-$Coef._n$
1 (u_r)	cos (nϕ)	sin (nϕ)
2 (u_ϕ)	sin (nϕ)	-cos (nϕ), = +1, n = 0
3 (u_t)	cos (nϕ)	sin (nϕ)
4 (θ_r)	sin (nϕ)	-cos (nϕ), = +1, n = 0
5 (θ_ϕ)	cos (nϕ)	sin (nϕ)
6 (θ_z)	sin (nϕ)	-cos (nϕ), = +1, n = 0

and

$$ID_n = \text{Ring ID} + (n \times 1000000). \tag{25}$$

6. SECTAX card (also processed in GEØM1 section)

-Input Card-

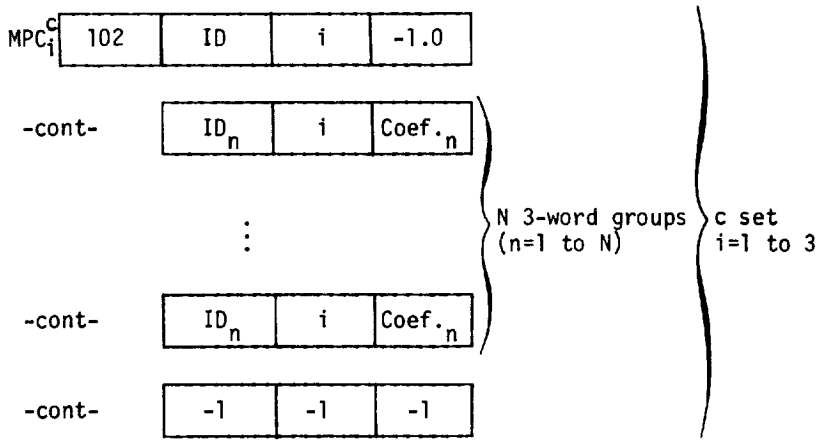
SECTAX	ID	Ring ID	R	ϕ_1	ϕ_2

-6 Output Cards-

MPC^s_i	101	ID	i	-1.0
-cont-	ID_n	i	$Coef._n$	
	⋮			
-cont-	ID_n	i	$Coef._n$	
-cont-	-1	-1	-1	

N 3-word groups (n=1 to N)

s set i=1 to 3



where the Coef.$_n$ above are defined by:

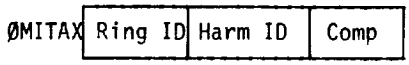
i	c set-Coef.$_n$	s set-Coef.$_n$
1	$\frac{R}{n}(\sin(n\phi_2) - \sin(n\phi_1))$ or $R(\phi_2 - \phi_1)$ for n=0	$-\frac{R}{n}(\cos(n\phi_2) - \cos(n\phi_1))$ or 0 for n=0
2	$-\frac{R}{n}(\cos(n\phi_2) - \cos(n\phi_1))$ or 0 for n=0	$-\frac{R}{n}(\sin(n\phi_2) - \sin(n\phi_1))$ or $R(\phi_2 - \phi_1)$ for n=0
3	$\frac{R}{n}(\sin(n\phi_2) - \sin(n\phi_1))$ or $R(\phi_2 - \phi_1)$ for n=0	$-\frac{R}{n}(\cos(n\phi_2) - \cos(n\phi_1))$ or 0, for n=0

and

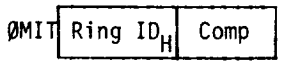
$$ID_n = \text{Ring ID} + (n \times 1000000) \tag{26}$$

7. ØMITAX card

-Input Card-



-Output card-



$$\text{Ring ID}_H = \text{Ring ID} + (\text{Harm ID} + 1) \times 1000000. \tag{27}$$

8. SUPAX card

-Input Card-

SUPAX	Ring ID	Harm ID	Comp

-Output Card-

SUPØRT	Ring ID_H	Comp

$$\text{Ring ID}_H = \text{Ring ID} + (\text{Harm ID} + 1) \times 1000000. \tag{28}$$

9. RINGAX card (also processed in GEØM1 section)

-Input Card-

RINGAX	Ring ID	R	Z	Comp

-2 x N Output Cards-

SPC_n^S	101	Ring ID_n^S	135	0

where

$$\text{Ring ID}_n^S = \text{Ring ID} + 1000000 \times n, \tag{29}$$

for n = 1, 2, ..., N; and

SPC_n^C	102	Ring ID_n^C	246	0

where

$$\text{Ring ID}_n^C = \text{Ring ID} + 1000000 \times n, \tag{30}$$

for n = 1, 2, ..., N.

4.6.7.5 Conversion of Input Bulk Data Cards to Output Cards for GEØM1.

1. PØINTAX card (also processed in GEØM4 section)

-Input Card-

PØINTAX	ID	Ring ID	ϕ

-Output Card-

GRID	ID	0	ϕ	0.0	0.0	0	0	0

2. SECTAX card (also processed in GEØM4 section)

-Input Card-

SECTAX	ID	Ring ID	R	ϕ_1	ϕ_2

-Output Card-

GRID	ID	0	R	ϕ_1	ϕ_2	0	0	0

3. RINGAX card (also processed in GEØM4 section)

-Input Card-

RINGAX	Ring ID	R	Z	Comp

-N Output Cards-

GRID	Ring ID_n	0	R	Z	0.0	0	Comp	0

where

$$\text{Ring ID}_n = \text{Ring ID} + 1000000 \times n, \tag{31}$$

for n = 1, 2, ..., N.

4.6.7.6 Order of Output for Generated Card Images

IFP3 has the responsibility to output cards in the sort order output by IFP. This causes IFP3 to simultaneously process some cards.

4.6.8 Subroutines

4.6.8.1 Subroutine Name: IFP3B

1. Entry Point: IFP3B
2. Purpose: To process the cards causing output to be created for GEØM4 and GEØM1.
3. Calling Sequence: CALL IFP3B

4.6.9 Design Requirements

The design requirements are:

1. To produce card images equivalent to those put out by IFP.
2. To output those images on GEØM1, GEØM2, GEØM3, GEØM4 in the proper sort and order.

The following CØMMØN blocks are required for data interface between subroutines IFP3 and IFP3B.

1. CØMMØN/IFP3LV/

   This CØMMØN block contains local variables common between IFP3 and IFP3B only.

2. CØMMØN/IFP3BD/

   This CØMMØN block contains constants common between IFP3 and IFP3B and is initialized in the Block Data subprogram AXICBD.

3. CØMMØN/IFP3ZZ/

   This CØMMØN block defines the beginning of open-core.

4.6.10 Diagnostic Messages

IFP3 error messages are all user-oriented. They pertain to Bulk Data card errors for the axisymmetric conical shell problem, and are output in summary form by IFP3 on the system output file.

4.7 EXECUTIVE PREFACE MODULE XGPI (EXECUTIVE GENERAL PROBLEM INITIALIZATION).

4.7.1 Entry Point: XGPI

4.7.2 Purpose

To translate (compile) a DMAP program into an internal form (the ØSCAR) for use by the NASTRAN Executive System, and, if restarting the problem, to initialize data blocks and named common blocks for proper restart of the problem. See section 2 for format of the ØSCAR.

4.7.3 Calling Sequence

CALL XGPI. XGPI is called only by subroutine SEMINT, the Preface driver.

4.7.4 Method

XGPI calls XGPIBS to initialize data for the module and to initialize the Link Specification table in named common block /XLINK/. Upon return from XGPIBS, XGPI loads the XCSA Executive Table into core from the Problem Tape. If restarting the problem, XGPI modifies table MEDMSK in named common block /XMDMSK/ if necessary. See discussion of the INM table in the description for the XCSA module, in section 4.2.6.2.

XGPI calls XØSGEN to execute phase 1 of the DMAP program compilation. XØSGEN processes the DMAP instructions and generates the skeleton of the Operation Sequence Control Array (ØSCAR). See section 2.4.2.1 for details on the format of the ØSCAR.

XGPI calls XFLØRD to execute phase 2 of the compilation. XFLØRD fills in the ØSCAR entries with the information needed for allocating files (by SFA) when DMAP modules are executed. If restarting a problem, XFLØRD determines which data blocks are needed from the Old Problem Tape to restart the problem and, when necessary, turns on execute flags for DMAP modules to regenerate missing data blocks.

At this point, XGPI terminates the job if any errors were found in compilation; if not, XGPI writes the ØSCAR onto the Data Pool File. If the problem is a restart, XGPI copies the data blocks specified by XFLØRD from the Old Problem Tape onto the Data Pool File and initializes various named common blocks.

XGPI calls ØSCDMP to print the ØSCAR if requested by the user via the DIAG card in the Executive Control Deck and to position the Data Pool File at the first ØSCAR entry in preparation for executing DMAP modules. If checkpointing is requested by the user, the Problem Tape Dictionary is initialized and written on the Problem Tape. XGPI then returns to the calling routine SEMINT.

4.7.5 Subroutines

The following labeled common blocks are used to communicate data and constants among the complex of XGPI subroutines.

1. CØMMØN/XGPIC/ - Contains 30 individual cells containing various flags, integer and BCD constants, and machine dependent data. Also an additional 40 cell array contains a series of required masks.

2. CØMMØN/XGPID/ - Contains restart type codes and approach type codes plus masks and flags required in ØSCAR generation.

3. CØMMØN/XGPI1/ - Defines the beginning of open core for the XGPI module and contains the ØSCAR as it is generated.

4. CØMMØN/XGPI2/ - Contains the MPL table (see section 2.4.2.2).

5. CØMMØN/XGPI2X/ - Contains the default parameters required by the MPL table.

6. CØMMØN/XGPI3/ - Contains the PVT table (see section 2.4.2.4) prior to it being written on the Problem Tape.

7. CØMMØN/XGPI4/ - Contains individual DMAP cards as they are output from XRCARD plus the various flags and pointers required to process each DMAP instruction.

8. CØMMØN/XGPI5/ - Contains solution, solution subset, approach and start codes along with data pertaining to DMAP ALTER numbers.

9. CØMMØN/XGPI6/ - Contains various pointers into the Module Execution Decision Table, MED (see section 1.10).

10. CØMMØN/XGPI7/ - Contains data pertaining to the IFILE Table (see section 4.7.6.3).

11. CØMMØN/XGPI8/ - Contains pointers into the ICPDPL Table (see section 4.7.6.3).

Further details regarding these common blocks may be obtained from the source listings for XGPIBD and XGPIBS.

4.7.5.1 XGPIDG

1. Entry Points: XGPIDG, XGPIMW.

2. Purposes: For XGPIDG, to write all fatal and non-fatal diagnostic messages for module XGPI. For XGPIMW, to write all non-diagnostic messages for module XGPI.

3. Calling Sequences:

For XGPIDG:

```
CALL XGPIDG (NCØDE,I,J,K)
```

NCØDE - Message code number

I,J,K - Integer values determined by NCØDE.

For XGPIMW:

```
CALL XGPIMW (MSGNØ,I,J,A)
```

MSGNØ - Message code number

I,J,A - Integer values determined by MSGNØ

4.7.5.2 XGPIBS

1. Entry Point: XGPIBS.

2. Purpose: To initialize module data and the Link Specification table in named common block /XLINK/ (see section 2.4).

3. Calling Sequence:

```
CALL XGPIBS
```

4.7.5.3 XØSGEN

1. Entry Point: XØSGEN.

2. Purpose: To execute phase 1 of the compilation by translating the DMAP program into a skeleton Operation Sequence Control Array (ØSCAR).

3. Calling Sequence:

```
CALL XØSGEN
```

4.7.5.4 XLNKHD

1. Entry Point: XLNKHD.

2. Purpose: To generate the header section of an ØSCAR entry and for problem restarts, to determine whether or not to set the ØSCAR entry execute flag.

3. Calling Sequence:

```
CALL XLNKHD
```

4.7.5.5 XIPFL

1. Entry Points: XIPFL, XØPFL.

2. Purpose: For XIPFL, to generate the input data block section of an ØSCAR entry. For XØPFL, to generate the output data block section of an ØSCAR entry.

3. Calling Sequences:

```
CALL XIPFL
CALL XØPFL
```

4.7.5.6 XPARAM

1. Entry Point: XPARAM.

2. Purpose: To generate the parameter section of an ØSCAR entry.

3. Calling Sequence:

```
CALL XPARAM
```

4.7.5.7 XSCNDM

1. Entry Point: XSCNDM.

2. Purpose: To scan all DMAP instructions and return to the calling program each item in an instruction along with its identification (i.e., delimiter, BCD name,

value or end of instruction) as it is requested.

3. Calling Sequence:

CALL XSCNDM

4.7.5.8 XFLØRD

1. Entry Point: XFLØRD.

2. Purpose: To compute the LTU (Last Time Used) and NTU (Next Time Used) values for the input and output sections of ØSCAR entries, and for problem restarts, to determine which data blocks are needed from the Old Problem Tape to restart the problem.

3. Calling Sequence:

CALL XFLØRD

4.7.5.9 XFLDEF

1. Entry Point: XFLDEF.

2. Purpose: To search the Old Problem Tape restart dictionary for a requested data block name and flag name if found; and if not found, and if restart is modified and the calling routine requests it, to attempt to regenerate the data block by turning on the proper ØSCAR execute flags.

3. Calling Sequence:

CALL XFLDEF (NAM1, NAM2, NØFIND)

NAM1,NAM2 - Data block name (8 characters, 4 characters/word).

NØFIND - For input, NØFIND < 0 indicates that the calling routine wants the data block regenerated if it is not in restart dictionary. NØFIND ≥ 0 indicates no regeneration is desired. For output, NØFIND indicates to the calling routine what XFLDEF did. NØFIND < 0, the data block was regenerated. NØFIND = 0, the data block was in the restart dictionary and was flagged for use in restarting the problem. NØFIND > 0, the data block was not found and was not regenerated.

4.7.5.10 ØSCDMP

1. Entry Point: ØSCDMP.

2. Purpose: To print the ØSCAR on the system output file if requested by user, and to position the Data Pool File at the first ØSCAR entry.

3. Calling Sequence:

CALL ØSCDMP (FILPØS)

FILPØS - The number of file marks to skip over in order to be positioned at beginning of the ØSCAR .

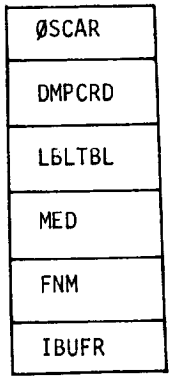
4.7.6 Design Requirements

4.7.6.1 Open Core Layout

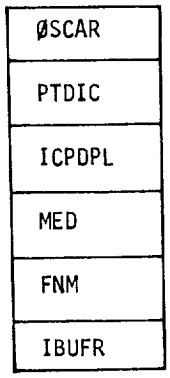
The ØSCAR array in named common block /XGPI1/ defines the first location in open core. All other arrays to be put in open core are equivalenced to ØSCAR and are offset from ØSCAR(1) by an amount determined at execution time. This dynamic allocation of arrays in open core optimizes the space available on a given machine, which means that any restrictions on data (except those noted below) are due to the machine's core size and not the program.

The diagrams below show the order in which tables reside in open core during phase 1 and phase 2 of the compilation.

Phase 1



Phase 2



In phase 1 the final sizes of the arrays ØSCAR and LBLTBL are not known until the DMAP program has been completely scanned by XØSGEN. These two arrays request space as needed until it runs out. At this time the user is informed that the DMAP should be shortened or

core storage should be increased.

4.7.6.2 Data Necessary For Operation

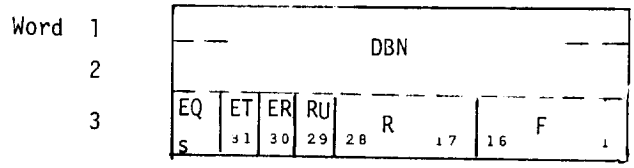
The Problem Tape provides Executive Tables XCSA, XALTER, PVT, and for restarts, XPTDIC (see section 2.4 for details). Data in named common blocks is initialized by the BLOCK DATA routines XMPLBD, XGPIBD and XBSBD or common block data is initialized by routine XGPIBS.

4.7.6.3 Table Formats

1. ØSCAR: Located in named common block /XGPI1/. See section 2.4.2.1 for format.

2. MED, CNM, FNM: Equivalenced to the ØSCAR table. See IS1, INM and JNM table descriptions in XCSA Module Functional Description (4.2.6.2).

3. PTDIC, ICPDPL: Equivalenced to ØSCAR array.
   Sample entry:



Word	Item	Description
1,2	DBN	Data block name (BCD) of the data block from the restart dictionary. Note, a data block name appears only once in the table except for table VPS where it appears twice.
3	R,F	Reel number and file number where the data block is last located on Old Problem Tape. For XVPS there is an entry (the first in PTDIC) which indicates where the first XVPS data block is located on the Old Problem Tape. For purged or not-generated data blocks, R = 0 and F = 0.
	EQ	Equivalence flag. EQ = 0 indicates the data block is equivalenced to another data block.
	ET	End of tape flag. ET = 1 indicates that the data block is split across two reels of the Old Problem Tape.

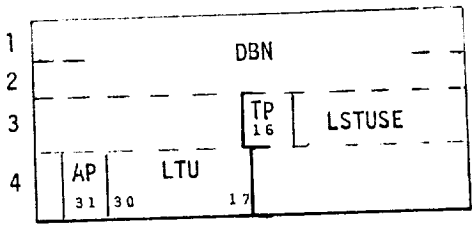
Word	Item	Description
	ER	End of logical record flag. ER = 1 indicates that the complete logical record was written out prior to changing reels when ET = 1.
	RU	Reuse flag. RU = 1 indicates that this data block is to be used to restart the problem.

Table ICPDPL contains all entries from PTDIC which had the RU flag set.

4. MPL: The MPL is located in named common block /XGPI2/. See section 2.4.2.2 for details

5. IØRDNL: Equivalenced to MPL, the IØRDNL table is used in phase 2 when the MPL is no longer needed. Data block names are entered into IØRDNL in the order that they are output from functional modules and IFP.

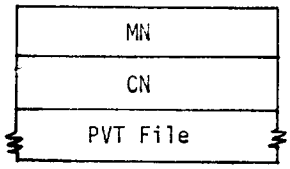
Sample entry:



Word	Item	Description
1,2	DBN	Data block name (BCD)
3	LSTUSE	Pointer to input or output section entry of a functional module ØSCAR entry where the data block was last referenced. LSTUSE is used to fill in NTU's (Next Time Used) in ØSCAR entries.
	TP	Tape flag. TP = 1 if the data block was declared TAPE in a FILE DMAP instruction.
	LTU	Last time used. Record number of ØSCAR entry beyond which the data block need not be saved for input.
	AP	Append flag. AP = 1 if the data block was declared APPEND in a FILE DMAP instruction.

6. PVT: Located in named common block /XGPI3/.

Sample entry:

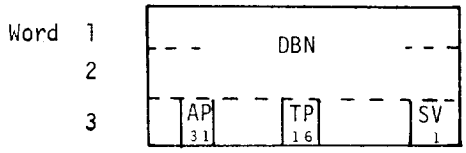


see section 2.4.2.4 for details.

MN - Maximum number of words in PVT (integer).

CN - Current number of words being used (integer).

7. IFILE: Located in named common block /XGPI7/. The purpose of IFILE is to save information from FILE DMAP instructions.



Word	Item	Description
1,2	DBN	Data block name (BCD)
3	SV	SAVE flag
	TP	TAPE flag
	AP	APPEND flag

4.7.6.4 Restrictions on Data

There are only three fixed length tables which might be overflowed by excess user data. These tables are PVT, IFILE and IØRDNL.

4.7.7 Diagnostic Messages

Every effort is made to detect syntactical and logical errors in the DMAP program, and, for restarts, to make sure that the problem is being restarted correctly. All tables are checked for overflow.

The NØGØ flag in named common block /SYSTEM/ is set according to the severity of the errors found. NØGØ = 1 indicates compilation is to be discontinued after phase 2. NØGØ = 2

indicates a serious error and causes XGPI to terminate the program immediately.

See the Diagnostic Message section of the User's Manual (section 6.2) for a detailed discussion of XGPI diagnostic messages. XGPI messages include numbers 1 thru 53.

4.8 EXECUTIVE PREFACE MODULE UMFEDIT (USER MASTER FILE EDITOR)

4.8.1 Entry Point: UMFEDT

4.8.2 Purpose

To create and manipulate User Master Files.

4.8.3 Calling Sequence

CALL UMFEDT. UMFEDT is called only by SEMINT, the Preface Driver.

4.8.4 Method

UMFEDIT functions as a post-processor to Executive Module XSØRT. Its primary task is to generate a User Master File by repeatedly transferring sorted bulk data decks generated by XSØRT from the New Problem Tape (NPTP) to the New User Master File (NUMF) based on control cards read from the System Input File. See section 2 of the User's Manual for a description of these control cards and how they control the contents of the NUMF.

In addition to creating a User Master File, UMFEDIT is used to list and/or punch Bulk Data Decks from an existing User Master File (UMF). Control cards read from the System Input File also control this process.

4.8.5 Subroutines

The UMFEDIT module has no auxiliary subroutines but uses XRCARD (see section 3.4 for a description).

4.8.6 Design Requirements

1. Open core is defined at /UMFXXX/ and is utilized as follows:

CØMMØN/UMFXXX/

GINØ Buffer for UMF
GINØ Buffer for UMF
GINØ Buffer for NPTP
: Unused core

2. The Block Data subprogram UMFZBD fills /UMFZZZ/.

3. UMFEDIT operates only in the Preface environment. The Bulk Data Deck must have been processed by XSØRT and accepted by the Input File Processor (IFP).

4.8.7 Diagnostic Messages

Bad Bulk Data Decks (indicated by ABØRT = .TRUE.) will not be accepted by UMFEDIT for inclusion on the NUMF. Subsequent Bulk Data Decks will be included, however, if acceptable.

Other errors detected in UMFEDIT will result in appropriate diagnostic messages being written on the System Output File and termination via PEXIT. These messages are: 1703 through 1716, 1718, 1719, 1721, 1722, 1723, and 1725 through 1737.

4.9 EXECUTIVE MODULE XSFA (EXECUTIVE SEGMENT FILE ALLOCATOR)

4.9.1 Entry Point: XSFA

4.9.2 Purpose

The Segment File Allocator (SFA) manages the data block to physical file relationships throughout a NASTRAN problem. Since, in general, the number of data blocks required for problem solution far exceeds the number of physical files available, allocation of files to data blocks is done dynamically as the module sequence proceeds. The SFA will allocate forward for as many modules as possible. A group of modules allocated by one operation of the SFA is termed a segment.

4.9.3 Calling Sequence

CALL XSFA (ØSCPØS)

ØSCPØS - When input to XSFA, this integer argument is the current position (record number) within the ØSCAR. Upon return from XSFA, (1) if allocation was successful, ØSCPØS is the ØSCAR position of the end of the segment as defined above; (2) if allocation was unsuccessful, the input argument is set negative.

4.9.4 Method

SFA is called by GNFIST (see section 3.3.9 for a description of GNFIST) when GNFIST fails to find the necessary data block names in the FIAT table to construct a complete FIST table for the next operating module (see section 2.4 for descriptions of the FIST and FIAT). The FIST table must contain an entry for each input, output, and scratch data block required by the module. SFA operates by processing the ØSCAR from its present position (next module to be operated) through all remaining modules. Only functional module and output processor ØSCAR entries flagged for execution are processed. The ØSCAR is read and processed by an internal subroutine named XSØSGN (Serial ØSCAR Sequence Generator) and the SØS table is formed. From this table all allocation is performed. Following XSØSGN, another internal subroutine named XCLEAN operates. XCLEAN acts to "clean up" the FIAT and DPL tables prior to the basic allocation procedure. This clean-up involves deleting data block names not needed for subsequent modules, removing equivalence flags if one member of an equivalent

pair is deleted, and closing up gaps in the FIAT caused by deletions. Following operation of XCLEAN, basic allocation begins. Allocation is accomplished during two passes through the SØS table.

Pass one first checks to see if each data block from the SØS is already in the FIAT. If so, it is considered allocated; otherwise the possibility of data block stacking is investigated. Stacking is defined as assigning two or more data blocks to the same physical file by considering their use span. The various use span attributes available are: First-Time-Used (FTU), Next-Time-Used (NTU), and Last-Time-Used (LTU). Data block A may use (be stacked on) the same file as data block B it the first use (FTU) of data block B is subsequent to the last use (LTU) of data block A. Thus many data blocks may be allocated to use the same physical file if their use spans do not overlap. Following pass one, if any data blocks within SØS remain un-allocated, pass two is begun.

Pass two first checks for files within the FIAT currently not assigned to any data block. These files are considered empty and are assigned to the un-allocated data blocks. Once the empty files are exhausted a check is made to determine if the next module to be operated has all its data blocks allocated to files. If the next module (at the least) is allocated, basic allocation is completed. If the next module has not been completely allocated, the second part of pass two will force pooling of sufficient data blocks to provide the necessary empty files for allocation of the remaining data blocks needed for the next module. Pooling is accomplished by flagging the data blocks for copying onto a separate file called the Data Pool File. The Data Pool File will therefore contain many different data blocks where all other files contain only one data block at a time. Data blocks are chosen for pooling by checking the next-time-used (NTU) attribute. The data block with the greatest NTU will be pooled first. Data blocks pooled are considered un-allocated; when they are subsequently re-allocated to their own file, they will be flagged for unpooling.

Following basic allocation, subroutine XPUNP (Pool-Unpool) is operated. All data blocks flagged for pooling are copied to the Pool File followed by all data blocks flagged for unpooling being copied from the Data Pool File. Lastly subroutine XDPH (Data Pool Housekeeper) operates to clean-up and if necessary re-copy the Data Pool File. SFA then returns control to GNFIST with the calling argument set negative if it was un-able to allocate the next module. Figure 1 illustrates the functional flow.

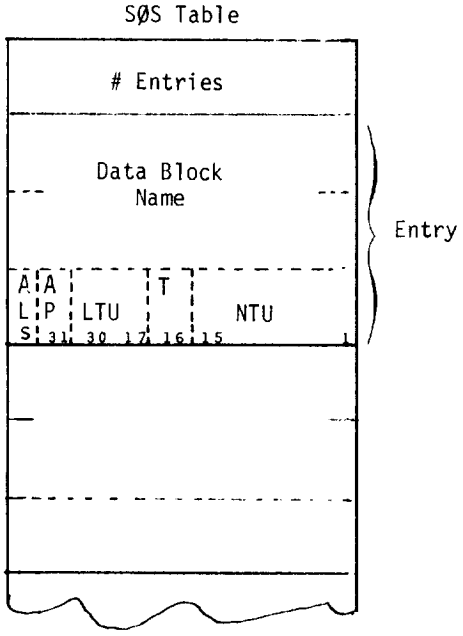
4.9.5 Subroutines

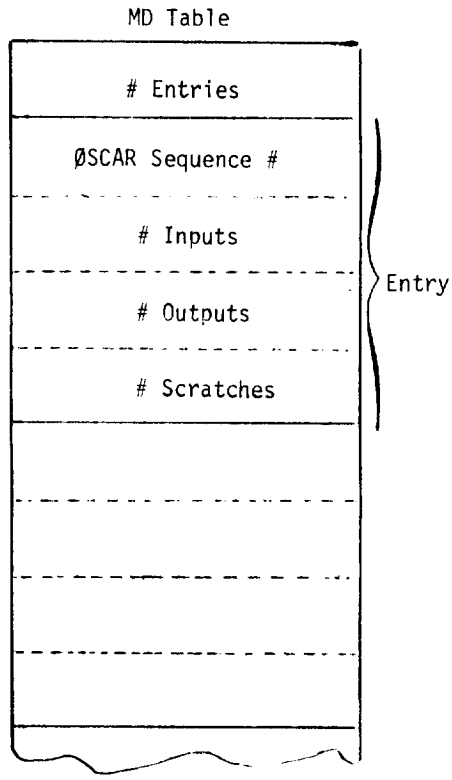
XSFA performs basic allocation and its entry point, purpose and calling sequence are given above. Below, it should be noted that XPLEQK and XFILPS are secondary entry points in XPØLCK.

4.9.5.1 Subroutine Name: XSØSGN (Serial ØSCAR Sequence Generator)

1. Entry Point: XSØSGN

2. Purpose: XSØSGN reads the ØSCAR and creates the SØS (Serial ØSCAR Sequence) and MD (Module Descriptor) tables. The SØS table contains the data block names and various attributes, while the MD table contains the ØSCAR sequence numbers and the number of input, output, and scratch data blocks required by each module.





AL = Allocation Flag, set on by SFA when data block is allocated.

AP = Append Flag, set on by module XGPI if data block is to be added to.

LTU = Last-Time-Used, created by XGPI as a data block attribute.

T = Tape Request Flag, set by XGPI to indicate a physical tape file is requested for data block.

NTU = Next-Time-Used, created by XSØSGN as a data block attribute.

= Full word integer values for items

Note: Items created or set by XGPI are passed via ØSCAR.

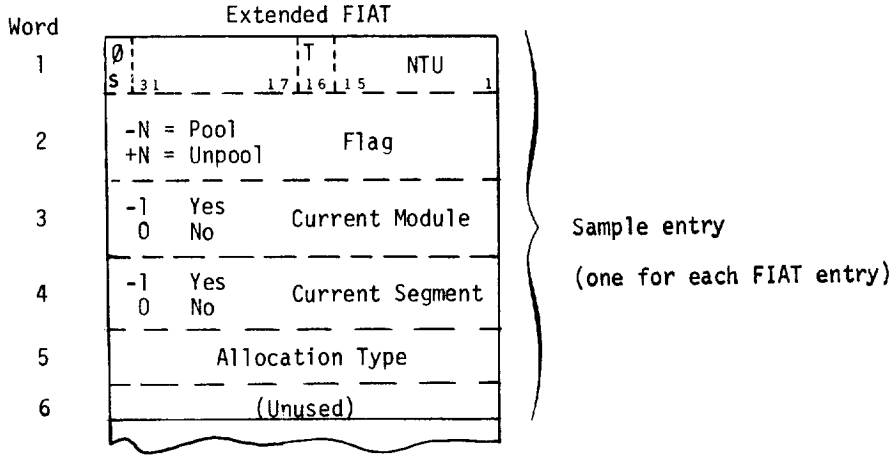
These tables are contained within the /XSFA1/ common block.

3. Calling Sequence: CALL XSØSGN

4.9.5.2 Subroutine Name: XCLEAN (FIAT and DPL Clean-up)

1. Entry Point: XCLEAN

2. Purpose: XCLEAN deletes data block names from the FIAT and DPL when they are no longer needed (their LTU has been reached). Following these deletions, equivalenced data blocks flags within FIAT are checked for continuing validity. If, for example, one member of an equivalenced pair has been deleted, the equivalence flag on the remaining member is removed. Finally, empty spaces within FIAT are removed by closing up the table. XCLEAN also regenerates and stores various parameters into the extended FIAT table. Since this table is non-resident and exists only during SFA operation, values such as NTU and the on/off switches must be restored.



Word	Item	Description
1	Ø	On/off switch, data block is turned off following stacking to prevent double stacking.
	T	Tape request flag, copied from SØS when allocated.
	NTU	Next-Time-Used; copied from SØS when allocated, regenerated by XCLEAN for data blocks remaining from previous allocation.

Word	Item	Description
2	-N	Flags XPUNP to pool data block, N = number of equivalent names for data block (Equals 1 if data block not equivalenced).
	+N	Flags XPUNP to unpool data block, N = Data Pool File number of data block.
3		Flag set yes if data block allocated for module currently being allocated - cleared between module allocations.
4		Flag set yes if data block allocated for module within current segment - i.e., all data blocks allocated during one SFA operation.
5		Allocation type - 1 = data block match, name already in FIAT 2 = data block stacking, name on file with another 3 = empty file used for data block 5 = data block using file freed by pooling another data block 7 = same as 5 except pooled data block is equivalenced
6		Unused

This table is contained within the /XSFA1/ common block.

4.9.5.3 Subroutine Name: XPUNP (Pool-Unpool)

1. Entry Point: XPUNP

2. Purpose: XPUNP checks the Pool and Unpool flags within the extended FIAT and performs the I/Ø operations necessary to copy data blocks from their separate files to the Data Pool File and vice-versa. Data block trailers are copied from the FIAT onto the Data Pool File as an additional record during pooling and are replaced from this record during unpooling. All requested pooling operations are performed prior to any unpooling operations. As data blocks are added to the Data Pool File, appropriate entries are added to the Data Pool Dictionary (DPL). The extended FIAT (4.9.5.2) is contained within the /XSFA1/ common block.

3. Calling Sequence: CALL XPUNP

4.9.5.4 Subroutine Name: XDPH (Data Pool Housekeeper)

1. Entry Point: XDPH

2. Purpose: XDPH scans the Data Pool Dictionary (DPL) to determine the number and size of the data blocks on the Data Pool File which have been flagged as no longer needed. If a sufficient quantity of data is flagged, a complete housekeeping operation is performed. This complete process involves re-copying the Data Pool File onto a scratch file while skipping those data blocks flagged for removal. Following the re-copy, the scratch file becomes the new Data Pool File and the old Data Pool File is released as a scratch file. XDPH will also perform a partial housekeeping if a (several) flagged data block(s) appears as the last data block on the Data Pool File. For this partial operation only the DPL entries are modified to release the pool space, no re-copy is necessary.

3. Calling Sequence: CALL XDPH

4.9.5.5 Subroutine Name: XPØLCK (Data Pool Dictionary Check)

1. Entry Point: XPØLCK

2. Purpose: XPØLCK scans the Data Pool Dictionary for a particular data block name. If found, the position within the dictionary and the Data Pool file number are returned to the calling program.

3. Calling Sequence: CALL XPØLCK (DBN1,DBN2,FN,L)

where:

DBN1, DBN2 - Request data block name (8 characters), 4 characters in each word, left justified and filled with blanks (if necessary).

FN - { 0, if data block not on the Data Pool File.
N, if data block on the Data Pool File; N = Data Pool File number.

L - Position of data block entry within the dictionary if data block found.

4.9.5.6 Subroutine Name: XPLEQK (Data Pool Equivalence Check)

1. Entry Point: XPLEQK

2. Purpose: XPLEQK scans the Data Pool Dictionary for any equivalence to the called data block name. If found, a copy of the equivalent names is moved from the Data Pool Dictionary to the FIAT.

3. Calling Sequence: CALL XPLEQK (PØØLX,FIATX)

where:

PØØLX - Position of data block entry within the dictionary (see argument L within XPØLCK).

FIATX - Position of same data block entry within the FIAT.

4.9.5.7 Subroutine Name: XFILPS (Data Pool File Positioner)

1. Entry Point: XFILPS

2. Purpose: XFILPS positions the Data Pool File to the beginning of a requested data block. (The Data Pool File is a multi-file file).

3. Calling Sequence: CALL XFILPS (FNEW)

where:

FNEW - The file count of the requested position. The current or old file position is stored in the /XSFA1/ common block.

4.9.6 Design Requirements

1. Open core is used only for GINØ buffers (2 maximum). The open core origin is common block /ESFA/ located following all SFA subroutines.

2. SFA communicates data internal to the module subroutines via common block /XSFA1/. The SØS, MD and extended FIAT tables reside in /XSFA1/.

3. The ØSCAR must be at the entry where allocation is to begin. Following allocation, that initial ØSCAR position is restored.

4. BLOCK DATA subprogram defines the lengths of tables in /XSFA1/.

4.9.7 Diagnostic Messages

Special DMAP routing may cause the warning 3022 message to be printed by SFA.

Following output of this message, SFA flags the data block as allocated (although it will not appear in the FIAT) and continues. Since SFA cannot predict conditional DMAP routing, the data block in question may not be required and the problem will proceed satisfactorily. If the data block is required, the problem will terminate in the requesting module. Under these circumstances the DMAP routing should be studied.

All other error messages generated by SFA are fatal in nature and indicate serious I/Ø malfunctions or executive table overflow. See section 6 of User's Manual for listing and explanation of these messages. XSFA messages include numbers 1001 through 1004, 1011 through 1014, 1021, 1031 through 1035, 1041, and 1051.

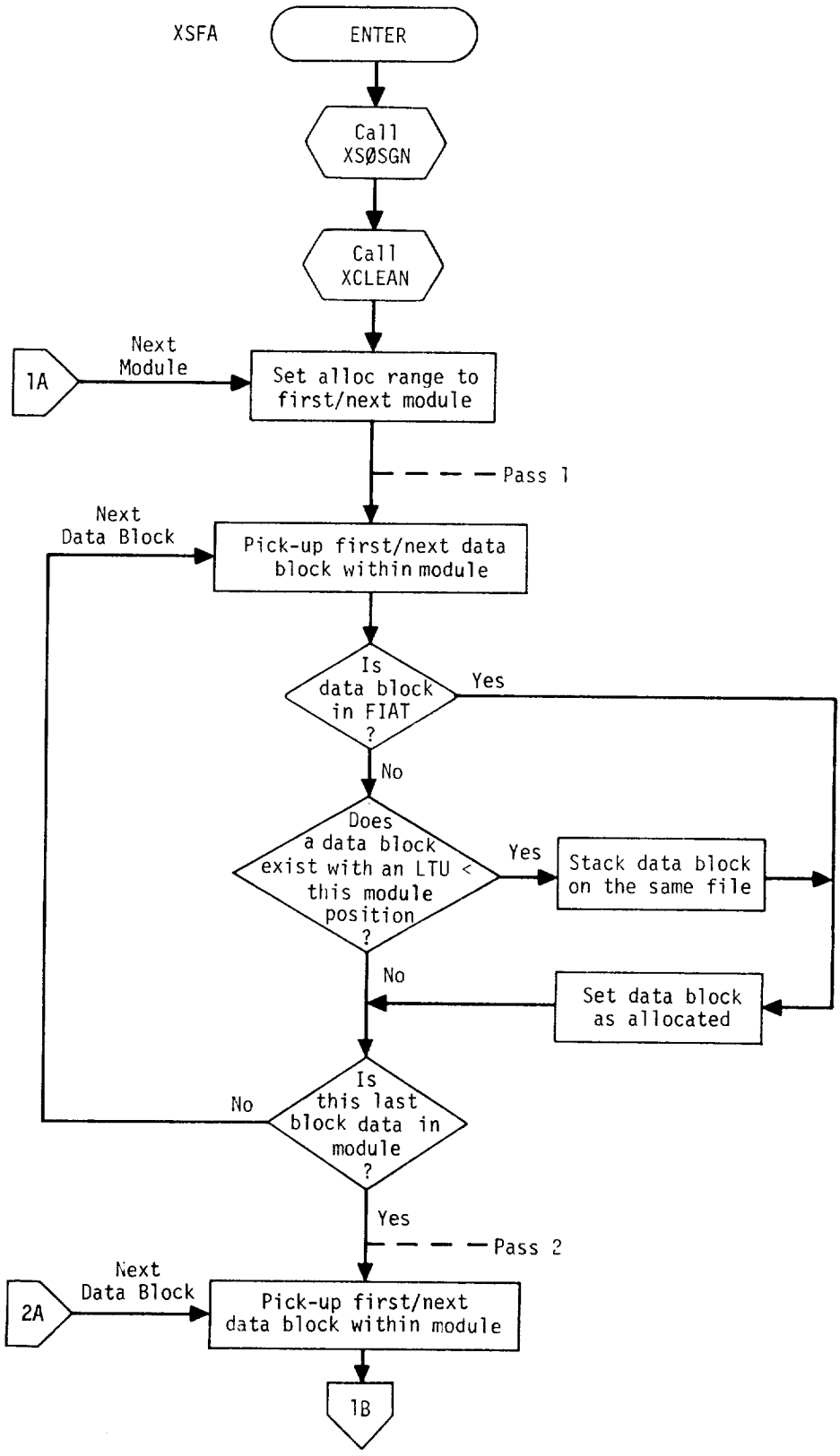


Figure 1.(a) Flowchart for module XSFA.

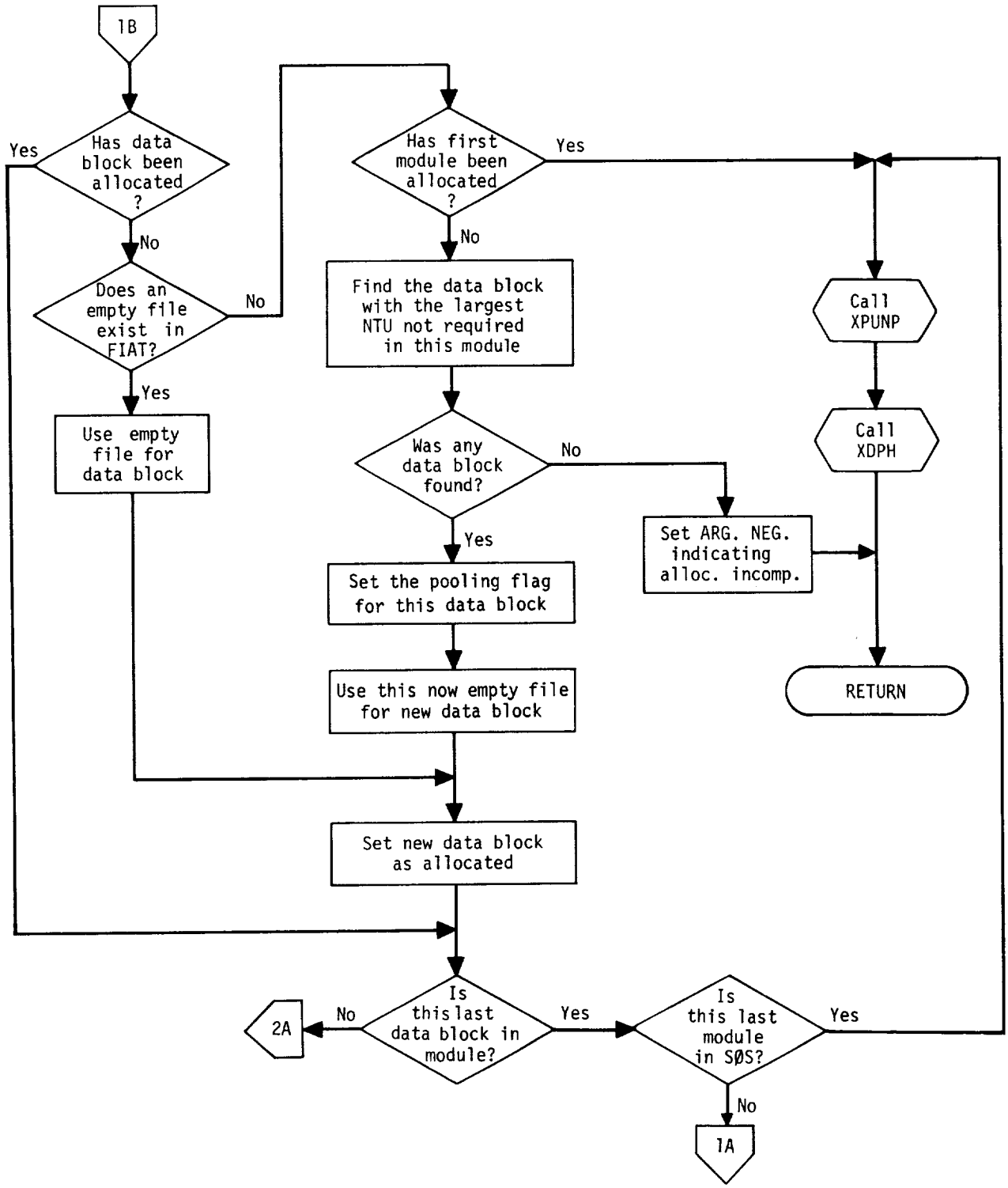


Figure 1.(b) Flowchart for module XSFA.

4.10 EXECUTIVE DMAP MODULE CHKPNT (CHECKPOINT)

4.10.1 Entry Point: XCHK

4.10.2 Purpose

To save, on the Problem Tape, specified data blocks along with other data necessary for restarting a problem.

4.10.3 DMAP Calling Sequence

CHKPNT DB1,DB2,...,DBN $

where DB1, DB2,...,DBN (N ≥ 1) are data blocks to be copied onto the Problem Tape for use in restarting a problem.

4.10.4 Method

The Problem Tape Dictionary (XPTDIC), see section 2.4.2.3, is brought into core from the Problem Tape, and the data block list in the CHKPNT ØSCAR entry is scanned. Data blocks that have been generated are entered into the local DICT table (see discussion below) along with all data blocks equivalenced to them. Data blocks that are purged, or have not yet been generated, are entered in the local PURGE table. The data blocks are assigned file numbers and are placed in the FDICT table. Data blocks are then copied onto the Problem Tape according to their file number unless the data block is equivalenced and is already in the Problem Tape Dictionary. Core resident data necessary for restart is also written on the Problem Tape as the VPS Executive Table. Entries from FDICT and PURGE are entered into the Problem Tape Dictionary, and the new checkpoint entries are punched for user submittal in the Executive Control Deck upon problem restart. The updated Problem Tape Dictionary is written back on the Problem Tape, and the Problem Tape is positioned to the beginning of the last file (i.e., XPTDIC) in preparation for the next execution of the CHKPNT module.

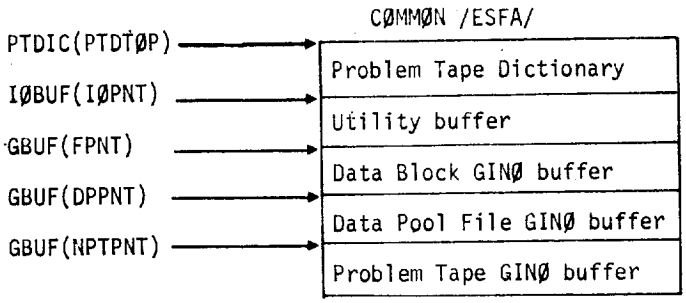
4.10.5 Subroutine

Module XCHK has no auxiliary subroutines.

4.10.6 Design Requirements

4.10.6.1 Open Core Layout

Named common block /ESFA/ defines the start of the open core area. The use of open core is optimized by origining arrays GBUF, PTDIC and IØBUF at /ESFA/ and computing their offset from the origin at execution time. The diagram below shows how the arrays are placed in open core.



4.10.6.2 Data Necessary for Operation

The data blocks, named common blocks and files needed by the CHKPNT module are listed below, along with type of access required (i.e. fetch and/or store data) and reasons for use.

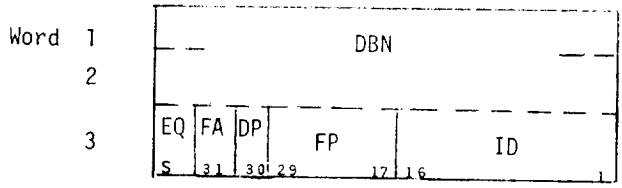
1. Data Pool File - fetch data blocks to be copied onto the Problem Tape.
2. Problem Tape - fetch and store data blocks.
3. Executive Table XPTDIC - fetch and store. Used to generate new checkpoint entries which are then added to XPTDIC.
4. Common /XFIST/ - store temporary entry in FIST for copying data blocks.
5. Common /ØSCENT/ - fetch. CHKPNT ØSCAR entry resides here.
6. Common /XCEITB/,/XVPS/ and /SYSTEM/ - fetch. Contains data to be written in VPS Executive Table.
7. Common /XFIAT/ and Common /XDPL/ - fetch. Contains data block names of data blocks to be copied.
8. Common /ØUTPUT/ - store. Prints out page heading for Checkpoint Dictionary printout.
9. Common /STAPID/ - fetch and store. Used to write Problem Tape ID file on new reel of of the Problem Tape when an end of reel is encountered.

4.10.6.3 Formats of Local Arrays

1. DICT table

The purpose of the DICT table is to hold preliminary data to be used in generating the FDICT table.

Sample DICT entry:

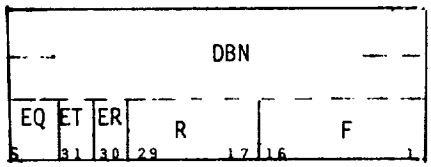


Word	Item	Description
1,2	DBN	BCD name (8 characters, 2 words) of the data block whose status is generated and is to be checkpointed.
3	EQ	Equivalence flag. EQ = 1 indicates all data blocks equivalenced to this data block also reside in DICT table.
	FA	File number assigned flag. FA = 1 indicates that an FDICT entry has been generated for this DBN and that a file number of where the data block will reside on the Problem Tape has been assigned.
	DP	Data pool flag. DP = 1 indicates that the data block is on the Data Pool File.
	FP	Varies according to DP flag. For DP = 1, FP is not used. For DP = 0, FP is a pointer to the FIAT entry containing DBN or equivalenced DBN.
	ID	Varies according to DP flag. For DP = 1, ID is the file number where the data block resides on the Data Pool File. For DP = 0, ID is the file (unit) identifier. For the IBM 7094, ID is the unit control block pointer for the file. For machines using the FØRTRAN GINØ, ID is the FØRTRAN logical unit number assigned to the file.

2. FDICT table

The purpose of the FDICT table is to hold final data to be used in generating XPTDIC entries for data blocks whose status is generated.

Sample FDICT entry:



Word	Item	Description
1,2	DBN	BCD name of data block.
	EQ	Equivalence flag. EQ = 1 indicates all data blocks equivalenced to this data block also reside in the FDICT table.
	ET	End-of-tape flag. ET = 1 indicates that the data block is split across two reels of Problem Tape.
	ER	End-of-logical-record flag. ER = 1 indicates that the complete logical record was written out prior to changing reels when ET = 1.
3	R,F	Reel number and file number where the data block will be written on the Problem Tape.

4.10.6.4 Restrictions

XPTDIC cannot be written across two reels of Problem Tape (i.e. a fatal error occurs if an end-of-tape is encountered while writing XPTDIC).

4.10.7 Diagnostic Messages

See Diagnostic Message section of User's Manual (section 6.2) for a detailed discussion of CHKPNT module diagnostic messages. XCHK messages include numbers 1101 through 1109.

4.11 EXECUTIVE DMAP INSTRUCTION REPT (REPEAT A GROUP OF DMAP INSTRUCTIONS)

4.11.1 Entry Point: XCEI

The XCEI module executes the DMAP control instructions: REPT, EXIT, CØND and JUMP.

4.11.2 Purpose

To repeat a group of DMAP instructions a specified number of times.

4.11.3 DMAP Calling Sequence

```
REPT  n,c $
```

where:

1. n is a BCD name appearing in a LABEL instruction which specifies the location of the beginning of the group of DMAP instructions to be repeated.

2. c is an integer constant which specifies the number of times to repeat the instructions.

4.11.4 Example

```
BEGIN  $
 .
 .
 .
LABEL  L1 $
MØDULE1  A/B/V,Y,P1 $
 .
 .
 .
MØDULEN B/C/V,Y,PN $
REPT  L1,3 $
 .
 .
 .
END $
```

The DMAP instructions from MØDULE1 to MØDULEN will be repeated 3 times. Note that REPT is placed at the end of the group of instructions to be repeated.

4.11.5 Method

Executive Table CEITBL in named common block /XCEITB/ (see section 2.4 for format) is searched for the REPT entry and the entry is updated after determining whether or not to repeat the loop again. If the loop is not to be repeated, a return is made to the calling routine. If a repeat of the loop is to be executed, the Problem Tape Dictionary (XPTDIC) is read into core from the Problem Tape, and dictionary entries created inside the loop are deleted. The updated XPTDIC is written back on the Problem Tape. Data blocks that are referenced only inside the loop and which have not been declared saved in a FILE DMAP instruction have their status changed to not-generated (i.e. data block trailers within FIAT are cleared and if the data block name appears in the DPL it is removed). The ØSCAR on the Data Pool Tape is positioned to the top of loop and a return is made to the calling routine.

4.11.6 Subroutine

4.11.6.1 Subroutine Name: XCEI.

1. Entry Point: XCEI

2. Purpose: To execute DMAP Control modules REPT, JUMP, CØND, and EXIT as described in the respective Executive DMAP Module Descriptions in this section and in sections 4.12, 4.13 and 4.14.

3. Calling Sequence:

```
CALL XCEI
```

4.11.7 Design Requrements

4.11.7.1 Open Core Layout

Named common block /ESFA/ defines the start of the open core area. The following diagram shows the layout.

CØMMØN/ESFA/

Data Pool File GINØ buffer area.
Problem Tape GINØ buffer area.
Problem Tape Dictionary (XPTDIC).

4.11.7.2 Data Necessary For Operation

The tables, named common blocks and files needed by the control modules are listed below, along with type of access required (i.e. fetch and/or store data) and reasons for use.

1. Data Pool File - XCEI must re-position the ØSCAR to the correct entry when a transfer is to be executed.

2. Problem Tape - fetch and store the Problem Tape Dictionary when looping.

3. Data Block XPTDIC - fetch and store. The Problem Tape Dictionary must be updated when looping.

4. CØMMØN /XVPS/ - fetch.

   /XVPS/ contains the CØND instruction parameter value.

5. CØMMØN /XCEITB/ - fetch and store.

   /XCEITB/ contains control parameters for REPT and EXIT instructions.

6. CØMMØN /ØSCENT/ - fetch.

   /ØSCENT/ contains the Control Module ØSCAR entry.

7. CØMMØN /XFIAT/ - fetch and store.

   Must be updated when looping.

8. CØMMØN /XDPL/ - fetch and store.

   Must be updated when looping.

4.12 EXECUTIVE DMAP INSTRUCTION JUMP (UNCONDITIONAL DMAP TRANSFER)

4.12.1 Entry Point: XCEI

4.12.2 Purpose

To alter the normal order of execution of DMAP modules by unconditionally transferring program control to a specified location in the DMAP program.

The normal order of execution of DMAP modules is the sequential order of occurrence of the modules as DMAP instructions in the DMAP program.

4.12.3 DMAP Calling Sequence

```
JUMP    n $
```

where n is a BCD name appearing on a LABEL instruction which specifies where control is to be transferred.

4.12.4 Method

If control is being transferred to a previous DMAP module in the ØSCAR (i.e., looping), the Problem Tape Dictionary is read into core from the Problem Tape and dictionary entries created inside the loop are deleted. The updated Problem Tape Dictionary is written back out on the Problem Tape. Data blocks that are referenced only inside the loop and which have not been declared saved in a FILE DMAP instruction have their status changed to not-generated. The ØSCAR is positioned to the specified location. Table CEITBL in named common block /XCEITB/ is searched for REPT entries and the loop count is zeroed if the jump is transferring control from inside the loop to outside the loop. A return is then made to the calling program.

If control is being transferred to a subsequent DMAP module in the ØSCAR, the ØSCAR is positioned to the specified location and a return is made to the calling program.

See description of the Executive DMAP instruction REPT (see section 4.11) for further details.

4.13 EXECUTIVE DMAP INSTRUCTION CØND (CONDITIONAL TRANSFER)

4.13.1 Entry Point: XCEI

4.13.2 Purpose

To alter the normal order of execution of DMAP modules by conditionally transferring program control to a specified location in the DMAP program.

4.13.3 DMAP Calling Sequence

CØND n,V $

where:

1. n is a BCD name appearing on a LABEL instruction which specifies where control is to be transferred.

2. V is a BCD name of a variable parameter whose value indicates whether or not to execute the transfer. If V < 0, the transfer is executed.

4.13.4 Example

```
BEGIN  $
  .
  .
  .
CØND  L1,K $
MØDULE1  A/B/V,Y,P1 $
  .
  .
  .
LABEL  L1 $
MØDULEN  X/Y/ $
  .
  .
  .
END  $
```

If K ≥ 0, MØDULE1 is executed. If K< 0 control is transferred to L1 and MØDULEN is executed.

4.13.5 Method

The parameter value for the CØND instruction is examined. If the value is greater than or equal to zero, a return is made to the calling routine. If the value is less than zero, the CØND instruction is executed exactly like the JUMP instruction. See description of the Executive DMAP instruction JUMP (section 4.12) for further details.

4.14 EXECUTIVE DMAP INSTRUCTION EXIT (TERMINATE DMAP PROGRAM)

4.14.1 Entry Point: XCEI

4.14.2 Purpose

To terminate a NASTRAN job.

4.14.3 DMAP Calling Sequence

```
EXIT  c $
```

where c is an integer constant which specifies the number of times the instruction is to be ignored before terminating the program. If c = 0 the calling sequence may be shortened to EXIT $.

4.14.4 Example

```
BEGIN $
 .
 .
 .
LABEL L1 $
MØDULE1 A/B/V,Y,P1 $
 .
 .
 .
EXIT 3 $
REPT L1,3 $
 .
 .
 .
END $
```

The EXIT instruction will be executed the third time the loop is repeated (i.e., the instructions within the loop will be executed four times).

4.14.5 Method

A determination is made whether or not to terminate the job by examining the loop count of the EXIT entry in named common block /XCEITB/. If the job is to be terminated, routine PEXIT is called; if not, the loop count in the EXIT entry is incremented, and a return is made to the calling program.

See description of the Executive DMAP instruction REPT (section 4.11) for further details.

4.15 EXECUTIVE DMAP MODULE SAVE (SAVE VARIABLE PARAMETER VALUES)

4.15.1 Entry Point: XSAVE

4.15.2 Purpose

To specify which variable parameter values are to be saved from the preceding functional module for use by subsequent modules.

4.15.3 DMAP Calling Sequence

```
SAVE    V1,V2,...,VN $
```

where V1,V2,...,VN (N ≥ 1) are the BCD names of some or all of the variable parameters which appear in the immediately preceding functional module DMAP instruction. A SAVE DMAP instruction must immediately follow the functional module instruction wherein the parameters being saved are generated.

4.15.4 Method

The specified parameter values are transferred from blank common to the VPS Executive Table located in named common block /XVPS/. See description of the ØSCAR in section 2.4.2.1 for the format of a SAVE ØSCAR entry.

4.15.5 Subroutine

The XSAVE module has no auxiliary subroutines

4.15.6 Design Requirements

SAVE must access blank common and named common blocks /XVPS/ and /ØSCENT/.

4.16 EXECUTIVE DMAP MODULE PURGE (EXPLICIT DATA BLOCK PURGE)

4.16.1 Entry Point: XPURGE

4.16.2 Purpose

To flag a data block so that it will not be allocated to a physical file and so that modules attempting to access it will be signaled.

4.16.3 DMAP Calling Sequence

```
PURGE  DBN1A,DBN2A,D3N3A/PARMA/DBN1B,DBN2B/PARMB  $
```

Note: The number of data block names (DBNiα) prior to each parameter (PARMα) and the number of sets of data block names and parameters in a particular calling sequence is variable.

4.16.4 Input Data Blocks

DBN1A,DBN2A, etc. - Any data block names appearing within the DMAP sequence.

4.16.5 Output Data Blocks

(None specified or permitted)

4.16.6 Parameters

PARMA, etc. - One required for each data block name or set of names.

4.16.7 Method

4.16.7.1 Summary

The data blocks (within the DMAP calling sequence) are purged if the value of the associated parameter is < 0. If the data blocks are already purged and the parameter value is ≥ 0, the purged data blocks are unpurged so that they may be subsequently reallocated. If the data blocks are not purged and if the parameter value is ≥ 0, no action is taken.

4.16.7.2 Functional Flow

PURGE operates by modifying entries within the FIAT (File Allocation Table) and DPL (Data Pool Dictionary). The FIAT contains an upper section (unique part) and a lower section (tail part). Both parts contain entries structured as described in the Executive Table description for the FIAT, section 2.4.1.2. The length of the unique part is defined by the unique files available count in the FIAT header. The tail part is defined as the remainder of the FIAT. The unique part contains one entry for each unique (separate) file available for allocation, and the file ID's within these entries are not modified through a NASTRAN run. The tail part contains entries for stacked files (see description for Executive Module XSFA, section 4.9) purged files, and members of equivalenced sets. An entry within the FIAT is purged by flagging (setting all bits on) its file ID. Therefore, if a data block within the unique part is to be purged, its name is moved to the tail. A data block entry within DPL is purged by removing its entry from the DPL. A data block which is already purged is unpurged by removing the flagged entry from the FIAT so that it may be subsequently allocated to a physical file. Figure 1 illustrates the logic flow.

4.16.8 Design Requirements

1. No open core is required by this module.

2. The ØSCAR record containing the DMAP purge request must reside in the labeled common block /ØSCENT/.

3. The validity of all data block names and controlling parameters is checked during NASTRAN initialization by module XGPI.

4.16.9 Diagnostic Messages

PURGE may produce the following System Fatal messages:

1201, FIAT ØVERFLØW

1202, DPL ØVERFLØW

Both of these messages indicate that the assembled size of the particular table has been exceeded. Although it is unlikely that either message will occur, a study of the erroneous problem's operation along with diagnostic prints of the FIAT and DPL, obtained via the DIAG Executive Control card (see User's Manual, section 2), should indicate some

corrective action. Possible corrective actions include: increasing the basic table size through re-assembly; providing more physical files to the NASTRAN system; and altering the DMAP operations.

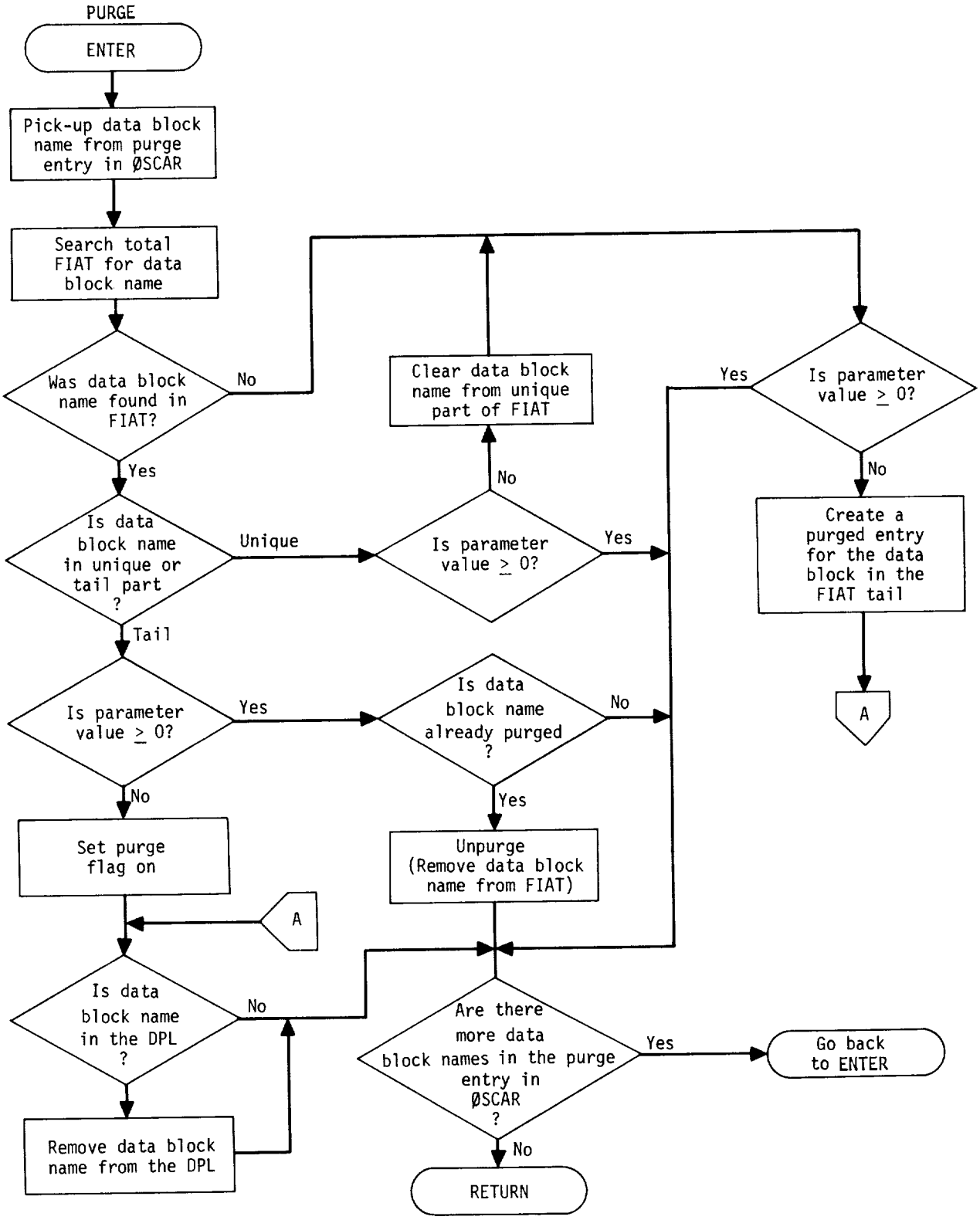


Figure 1. Flowchart for module PURGE.

4.17 EXECUTIVE DMAP MODULE EQUIV (DATA BLOCK NAME EQUIVALENCE)

4.17.1 Entry Point: XEQUIV

4.17.2 Purpose

To attach one or more equivalent (alias) data block names to an existing data block so that module accesses to data by equivalenced names will be identical.

4.17.3 DMAP Calling Sequence

```
EQUIV  DBN1A,DBN2A,DBN3A/PARMA/DBN1B,DBN2B/PARMB  $
```

Note: The number of data block names (DBNiα) prior to each parameter (PARMα) and the number of sets of data block names and parameters in a particular calling sequence are variable.

4.17.4 Input Data Blocks

DBN1A,DBN2A, etc. - Any data block names appearing within the DMAP sequence. The 1st data block name in each series (DBN1A and DBN1B) is primary and the 2nd, etc. data block names become equivalent to the primary.

4.17.5 Output Data Blocks

(None specified or permitted)

4.17.6 Parameters

PARMA, etc. - One required for each set of data block names.

4.17.7 Method

4.17.7.1 Summary

The data block names are made equivalent if the value of the associated parameter is < 0. If a set of data blocks is already equivalenced and the parameter value is ≥ 0, the equivalence is broken and the data block names again become unique. If the data blocks are not equivalenced and if the parameter value is ≥ 0, no action is taken.

4.17.7.2 Functional Flow

EQUIV operates by modifying entries within the FIAT (File Allocation Table) and DPL (Data Pool Dictionary). The FIAT contains an upper section (unique part) and a lower section (tail part). Both parts contain entries structured as described in the Executive Table description for the FIAT, section 2.4. The length of the unique part is defined by the unique files available count in the FIAT header. The tail part is defined as the remainder of the FIAT. The unique part contains one entry for each unique (separate) file available for allocation and the file ID's within these entries are not modified through a NASTRAN run. The tail part contains entries for stacked files (see description for Executive Table XSFA), purged files and members of equivalenced sets. Entries within the FIAT and DPL are made equivalent by setting their EQUIV flags (sign bit within an entry) and making their file ID's identical. Since a data block within the unique part of the FIAT must have a unique file ID, only one member of an equivalence set may reside within the unique section, all others will be placed in the FIAT tail. Thus, if two data blocks occupying unique physical files are equivalenced, one will be moved to the FIAT tail. Data blocks previously equivalenced are unequivalenced (broken) by removing the EQUIV flags and the secondary entries. When two or more data blocks are equivalenced, the first data block of the set is considered the primary data block. All others are considered secondary. The file containing the primary data block is logically attached to all data blocks in the set: primary and secondary. Data on files attached to secondary data blocks prior to equivalencing is lost upon equivalence. If the primary data block is purged, the secondary(s) will be purged. Figure 1 illustrates the logic flow.

4.17.8 Design Requirements

1. No open core is required by the module.

2. The ØSCAR record containing the DMAP EQUIV request must reside in the labeled common block /ØSCENT/.

3. The validity of all data block names and controlling parameters is checked during NASTRAN initialization by XGPI.

4. XEQUIV is an entry point in XPURGE.

4.17.9 Diagnostic Messages

EQUIV may produce the following System Fatal Messages:

1201 FIAT ØVERFLØW

1202 DPL ØVERFLØW

Both of these messages indicate that the assembled size of the particular table has been exceeded. Although it is unlikely that either message will occur, a study of the erroneous problem's operation along with diagnostic prints of the FIAT and DPL obtained, via the DIAG Executive Control card (see User's Manual, section 2), should indicate some corrective action. Possible corrective actions include: increasing the basic table size through re-assembly; providing more physical files to the NASTRAN system; and altering the DMAP operations.

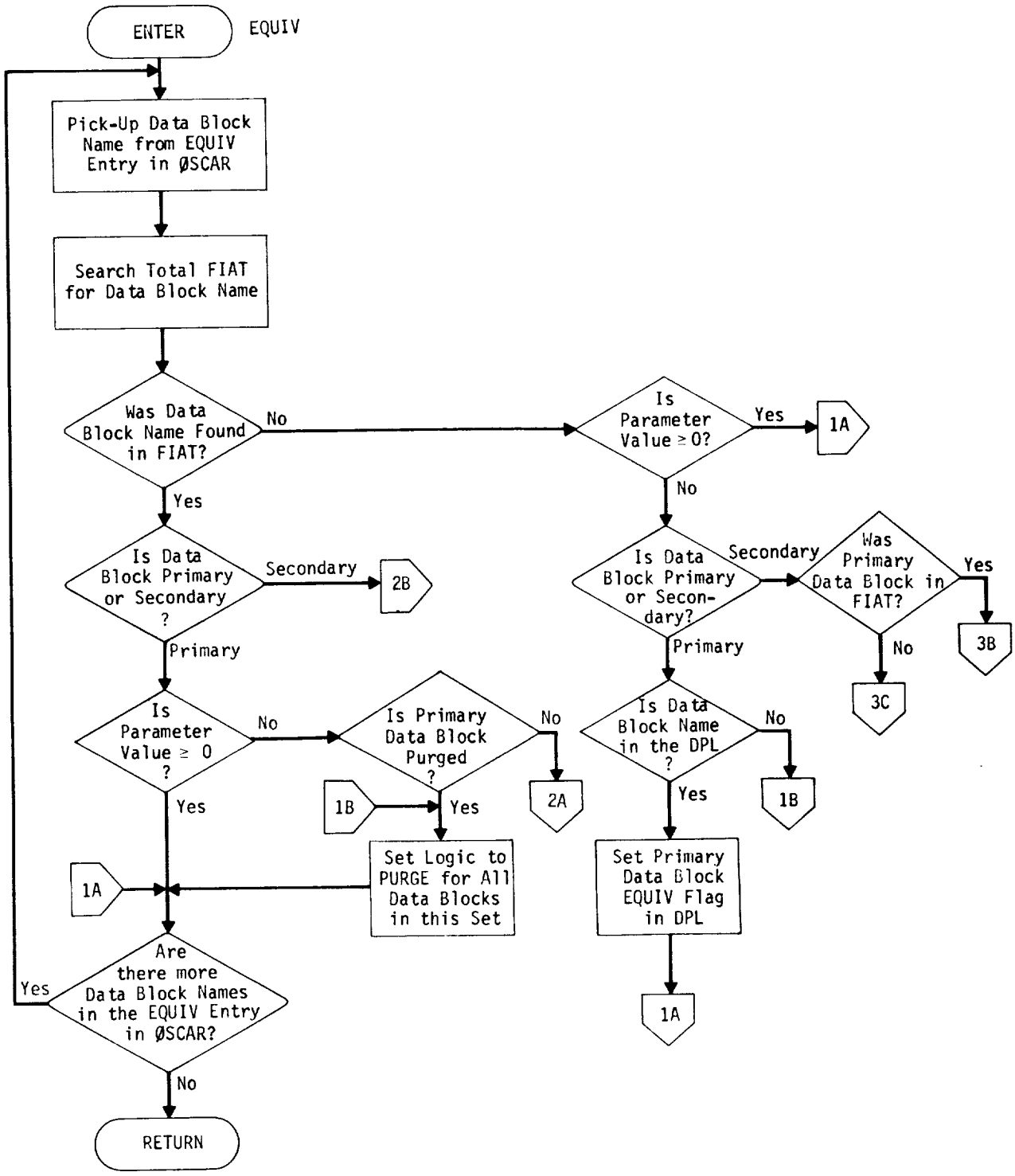


Figure 1. (a) Flowchart for EQUIV module

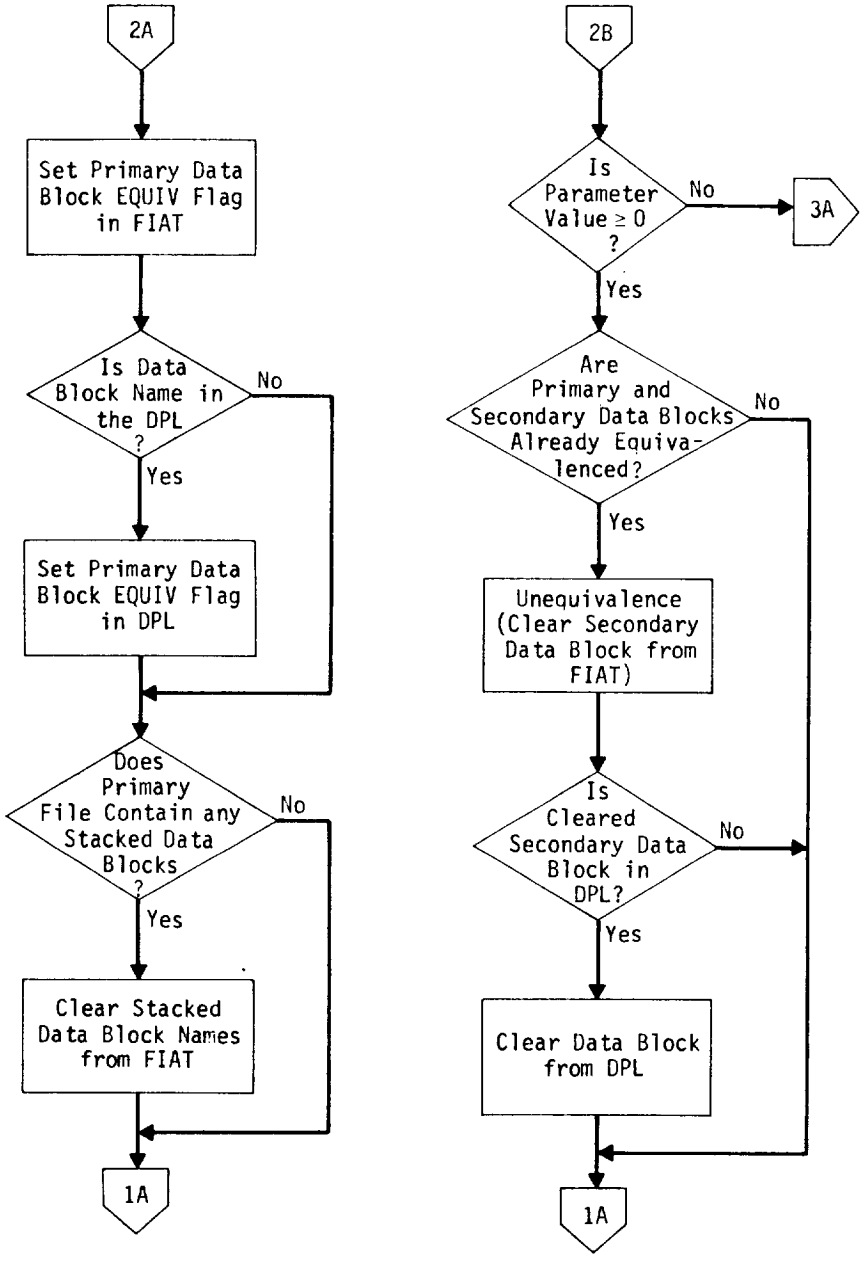


Figure 1. (b) Flowchart for EQUIV module

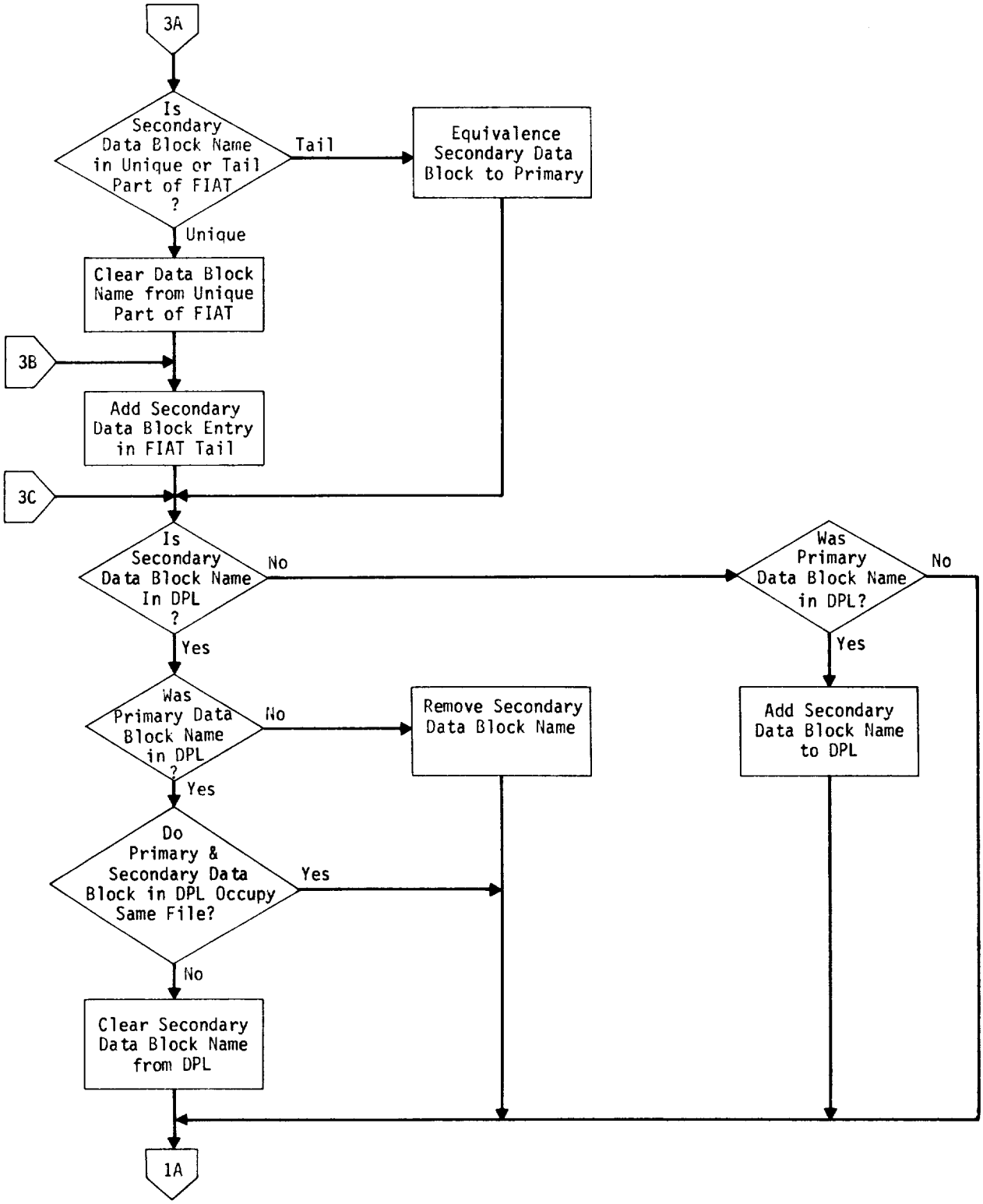


Figure 1. (c) Flowchart for EQUIV module

4.18 EXECUTIVE DMAP INSTRUCTION END (END OF DMAP PROGRAM)

4.18.1 Entry Point: XCEI

The XCEI module executes the DMAP control instructions: REPT, EXIT, CØND, and JUMP.

4.18.2 Purpose

To denote the end of a DMAP program. This DMAP instruction performs a function similar to an END statement in a FØRTRAN compilation, i.e., to signal the end of the source program.

4.18.3 DMAP Calling Sequence

```
END   $
```

Note: An END DMAP instruction is operationally equivalent to an EXIT 0 $ or EXIT $ DMAP instruction.

4.18.4 Method

The END instruction is translated during a DMAP program compilation in module XGPI into an EXIT $ instruction. (see section 4.14).

4.19 EXECUTIVE DMAP MODULE PARAM (PARAMETER PROCESSOR)

4.19.1 Entry Point: QPARAM

4.19.2 Purpose

To perform specified arithmetic and logical operations on DMAP parameters.

4.19.3 DMAP Calling Sequence

PARAM //C,N,ØP/V,N,ØUT/V,N,IN1/V,N,IN2 $

where the following operations (ØP) are available:

ØP	ØUT	IN1	IN2	
AND	-1	< 0	< 0	
	+1	< 0	≥ 0	
	+1	≥ 0	< 0	
	+1	≥ 0	≥ 0	
ØR	-1	< 0	< 0	
	-1	< 0	≥ 0	
	-1	≥ 0	< 0	
	+1	≥ 0	≥ 0	
ADD	IN1+IN2			Integer Arithmetic
SUB	IN1-IN2			
MPY	IN1*IN2			
DIV	IN1/IN2			
NØT	-IN1		*	
IMPL	-1	< 0	< 0	
	+1	< 0	≥ 0	
	-1	≥ 0	> 0	
	-1	≥ 0	≥ 0	
NØP	ØUT	*	*	

Notes:

1. *not used.

2. PARAM does its own SAVE; therefore, a DMAP SAVE instruction is not needed following the module.

3. PARAM has no input or output data blocks.

4.19.4 Examples

1. PARAM //C,N,NØP/V,N,P1=5 $ - this example sets the value of parameter P1 to 5 and saves it in the VPS.

2. PARAM //C,N,NØT/V,N,XYZ/V,N,NØXYZ $ - this example changes the sense of parameter NØXYZ which may be useful for the CØND or EQUIV instructions. Alternatively, XYZ could have been set in the following way:

```
PARAM     //C,N,MPY/V,N,XYZ/V,N,NØXYZ/C,N,-1   $
```

4.19.5 Method

QPARAM performs the indicated parameter operation and stores the result in the VPS (/XVPS/).

4.19.6 Diagnostic Messages

ØPERATIØN CØDE _____ NØT DEFINED FØR MØDULE PARAM. EXECUTIØN TERMINATED.

The listed operation code was not recognized by PARAM.

4.20 EXECUTIVE DMAP MODULE SETVAL (SET VALUES)

4.20.1 Entry Point: SETVAL

4.20.2 Purpose

To set DMAP parameters equal to other DMAP parameters or to constants.

4.20.3 DMAP Calling Sequence

```
SETVAL    //V,N,X1/V,N,Y1/V,N,X2/V,N,Y2/V,N,X3/V,N,Y3/V,N,X4/V,N,Y4/V,N,X5/V,N,Y5 $
```

4.20.4 Input Data Blocks

None.

4.20.5 Output Data Blocks

None.

4.20.6 Parameters

X1, X2, X3, X4, X5 - Output-integers-no default values.

Y1, Y2, Y3, Y4, Y5 - Input-integers-default values = -1.

4.20.7 Method

This module does nothing except set X1 = Y1, X2 = Y2, X3 = Y3, X4 = Y4, and X5 = Y5. Only two parameters need be specified in the calling sequence (X1 and Y1).

4.20.8 Subroutines

SETVAL has no auxiliary subroutines.

4.20.9 Design Requirements

SETVAL should reside in the root segment in all links.

4.20.10 Diagnostic Messages

None.

4.21 FUNCTIONAL MODULE GP1 (GEOMETRY PROCESSOR - PHASE 1)

4.21.1 Entry Point: GP1.

4.21.2 Purpose

GP1 performs basic geometry processing for the model. A list of all grid and scalar points is assembled and placed in internal order. Coordinate system transformation matrices are computed, and all grid points are transformed to the basic coordinate system.

4.21.3 DMAP Calling Sequence

```
GP1    GEØM1,GEOM2/GPL,EQEXIN,GPDT,CSTM,BGPDT,SIL/V,N,LUSET/V,N,NØCSTM/V,N,NØGPDT $
```

4.21.4 Input Data Blocks

GEØM1 - Grid point, coordinate system, sequence data.

GEØM2 - Element connection data.

4.21.5 Output Data Blocks

GPL - Grid Point List.

EQEXIN - Equivalence between external grid or scalar numbers and internal numbers.

GPDT - Grid Point Definition Table.

CSTM - Coordinate System Transformation Matrices.

BGPDT - Basic Grid Point Definition Table.

SIL - Scalar Index List.

Note: No output data block may be purged.

4.21.6 Parameters

LUSET - Output, integer, no default. Total degrees of freedom in the g displacement set.

NØCSTM - Output, integer, no default. Number of coordinate systems defined in the Bulk Data Deck, -1 if no coordinate systems defined.

NØGPDT - Output, integer, no default. -1 if no grid or scalar points defined in Bulk Data Deck, +1 otherwise.

4.21.7 Method

4.21.7.1 Construction of the GPL and First Logical Record of the EQEXIN.

The SPØINT cards and the scalar element cards (CELASi, CDAMPi, CMASSi, i = 1,2,3,4) are read from GEØM2, and a list is made of all referenced scalar points. The GRID cards are read from GEØM1, and a merged list of all grid and scalar points is constructed and written on SCR1, a scratch file. The list is expanded to pairs of numbers. The first number is the identification number, ID, the second is the resequenced number which is given on the SEQGP cards or is 1000*ID if not given on SEQGP cards. The paired list is sorted by SØRT on the sequence numbers. The resulting set of first numbers is written as the first logical record in the GPL (Grid Point List). These are the point identification numbers in order of their sequence numbers. The sequenced paired list is written as the second logical record of the GPL data block. The second numbers in the sequenced paired list are replaced by the indices 1, 2, 3,..., according to position. The list is sorted again, this time using the first number of each pair (the identification number). The resulting paired list is the first logical record of the EQEXIN data block which is used to convert external numbers, given by the first number of a pair, to the internal grid point indices, given by the second number in the pair.

4.21.7.2 Formats of GPDT, BGPDT and CSTM.

The geometry data blocks are the GPDT, the BGPDT and the CSTM. Their formats, although described in section 2.3.3, are repeated here since the following terms will be referenced in the discussion below on the construction of the CSTM.

GPDT - There is one entry for each grid or scalar point. The order of the entries is by the internal (sequenced) order. Each entry contains:

1. Internal sequence number.
2. Locating coordinate system ID.
3. x,y,z for a rectangular system.
4. r,θ,z for a cylindrical system.
5. ρ,θ,ϕ for a spherical system.
6. Global coordinate local coordinate system ID.
7. Permanent single-point constraint coordinate (1 = x, 2 = y, etc.).

For scalar points, word 2 = -1 and words 3 through 7 are zero. The data is essentially a duplicate of the GRID bulk data card except that the identification number is replaced by the internal sequence number.

. BGPDT - Contains one entry for each grid or scalar point. The contents are:

1. Local coordinate system ID for global coordinate definition.
2. x_i } Locations of point
3. y_i } in basic coordinate
4. z_i } system.

CSTM - The CSTM contains one entry for each local coordinate system. The order is by coordinate system identification numbers. Each entry contains 14 words:

Word	Item
1	N - the coordinate system ID.
2	$Type_N$ - the coordinate system type (rectangular, cylindrical or spherical).
3-5	$\{R_{ON}\}$ - the location of the system origin in basic coordinates.
6-14	$[T_{ON}]$ - the three-by-three matrix defining the orientation of the coordinate system principal axes.

4.21.7.3 Construction of the GPDT.

The GPDT data block is formed in core sized groups. The grid and scalar data are read one entry at a time from SCR1. EQEXIN (in core) is searched to find the internal number, and the grid data are stored (if possible) in the internal position allocated in core. If core will not hold the GPDT, the data are written on SCR2, and SØRT is called to sort and write the data on the GPDT.

4.21.7.4 Construction of the CSTM.

Sixteen words are alloted for each local coordinate system, and five words are alloted for each referenced grid point. The CØRDij data is read from GEØM1 and stored in core. External point ID's on CØRDij cards are replaced with internal numbers. A CØRD1j card references three grid points. It may be converted to a CSTM entry if these grid points have their locations reduced to basic coordinates. A CØRD2j card references another local coordinate system. It may

be converted to a CSTM entry if that referenced system has been reduced to a CSTM entry. The basic logic is to make repeated passes over the coordinate system data, each time reducing one or more coordinate systems and, when possible, converting referenced grid points to basic system location.

A CØRD1j card image references three grid points - a, b and c. If the locations of these points in basic coordinates are the vectors $\{R_a\}$, $\{R_b\}$, $\{R_c\}$, the solution for coordinate system N is

$$\{R_{ON}\} = \{R_a\} , \tag{1}$$

$$\{V_k\} = \{R_b\} - \{R_a\}, \tag{2}$$

$$\{V_i\} = \{R_c\} - \{R_a\} , \tag{3}$$

$$\{k\} = \frac{\{V_k\}}{|\{V_i\}|} \text{ (unit "z" vector)}, \tag{4}$$

$$\{j\} = \frac{\{k\} \times \{V_i\}}{|\{k\} \times \{V_i\}|} \text{ (unit "y" vector)}, \tag{5}$$

$$\{i\} = \{j\} \times \{k\} \text{ (unit "x" vector)}. \tag{6}$$

Point a is the origin, point b lies on the z (or polar) axis, point c lies in the x-z plane ($\theta = 0$ or $\phi = 0$). The three-by-three matrix $[T_N]$ is defined as:

$$[T_N] = \begin{bmatrix} i_1 & j_1 & k_1 \\ i_2 & j_2 & k_2 \\ i_3 & j_3 & k_3 \end{bmatrix} . \tag{7}$$

N, $type_N$, $\{R_{ON}\}$ and $[T_N]$ form the CSTM entry for the coordinate system.

A GRiD point (j) referenced to coordinate system (N) may be reduced to basic coordinates (X_o, Y_o, Z_o) by the equations:

1. If $type_N$ = Rectangular (R), X_j, Y_j and Z_j are given by

$$\begin{Bmatrix} X_o \\ Y_o \\ Z_o \end{Bmatrix} = \{R_{ON}\} + [T_N] \begin{Bmatrix} X_j \\ Y_j \\ Z_j \end{Bmatrix}. \tag{8}$$

2. If $type_N$ = Cylindrical (C), r, θ and Z are given by

$$X_j = r \cos \theta, \tag{9}$$

$$Y_j = r \sin \theta, \tag{10}$$

$$Z_j = Z. \tag{11}$$

X_o, Y_o and Z_o are calculated as in Equation 8.

3. If $type_N$ = Spherical (S), ρ, θ and ϕ are given by

$$X_j = \rho \sin \theta \cos \phi, \tag{12}$$

$$Y_j = \rho \sin \theta \sin \phi, \tag{13}$$

$$Z_j = \rho \cos \theta. \tag{14}$$

X_o, Y_o and Z_o are calculated as above.

When the basic location of a grid point has been calculated, the entry in the list is changed such that the reference coordinate system (entry No. 2) is zero and the three values are X_o, Y_o, Z_o.

The CØRD2j card image references another coordinate system and defines three points in the referenced system: a, b and c. If system number N is defined by system number M, the solution is

1. If $type_M$ = rectangular, the numbers defining the three points are the vectors: {a}, {b}, and {c}.

2. If $type_M$ = cylindrical, the numbers are r, θ and z. The equations to convert these to rectangular vectors are

$$\begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} r_a \cos \theta_a \\ r_a \sin \theta_a \\ Z_a \end{Bmatrix} = \{a\} . \tag{15}$$

The {b} and {c} vectors are calculated similarly.

3. If $type_M$ = spherical, the numbers given for the points are ρ, θ, and φ. We calculate

$$\begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} a\rho \sin \theta \cos \phi \\ a\rho \sin \theta \sin \phi \\ a\rho \cos \theta \end{Bmatrix} = \{a\} \tag{16}$$

and similarly for points b and c.

4. The definition of the new system is that point a is the origin, point b lies on the z (or polar) axis and point c lies in the x-z (or θ = 0) plane. The equations for the CSTM data are

$$\{R_{ON}\} = \{R_{OM}\} + [T_{ON}] \{a\}. \tag{17}$$

In system M the vectors defining the axes of system N are

$$\{V_k\} = \{b\} - \{a\}; \tag{18}$$

$$\{k\} = \frac{\{V_k\}}{|\{V_k\}|} , \text{("z" unit vector)}; \tag{19}$$

$$\{V_i\} = \{c\} - \{a\}; \tag{20}$$

$$\{j\} = \frac{\{k\} \times \{V_1\}}{|\{k\} \times \{V_1\}|} \text{ , ("Y" unit vector);} \tag{21}$$

$$\{i\} = \{j\} \times \{k\} \text{ , ("X" unit vector).} \tag{22}$$

The orientation of the axes is defined by the matrix

$$[T_{ON}] = [T_{OM}] \begin{bmatrix} i_1 & j_1 & k_1 \\ i_2 & j_2 & k_2 \\ i_3 & j_3 & k_3 \end{bmatrix} . \tag{23}$$

5. On each pass of the CØRDij data at least one new system must be converted. After each pass the referenced GRID data is checked and converted. The resulting CØRDij data will be the CSTM data block with each entry reduced from 16 to 14 words.

4.21.7.5 Construction of the BGPDT, the SIL and the Second Logical Record of the EQEXIN.

The BGPDT and the SIL data blocks are formed simultaneously. The SIL data block is simply a list of the first scalar index for each grid or scalar point. A grid point has six scalar indices (or degrees of freedom), and a scalar point has one scalar index. Every degree of freedom in the problem has a scalar index, but since the six degrees of freedom for a grid point are consecutive, only the first one is listed.

The GPDT data are read a point at a time. The basic location coordinates of the point are formed using Equation 8 through Equation 14 and these data are written on the BGPDT file. The SIL value for the next point is calculated by incrementing the last value by six (grid point) or by one (scalar point).

Finally the second logical record of EQEXIN is written. This record contains pairs of external numbers, 10*scalar index + type where type = 1 for a grid point, 2 for a scalar point.

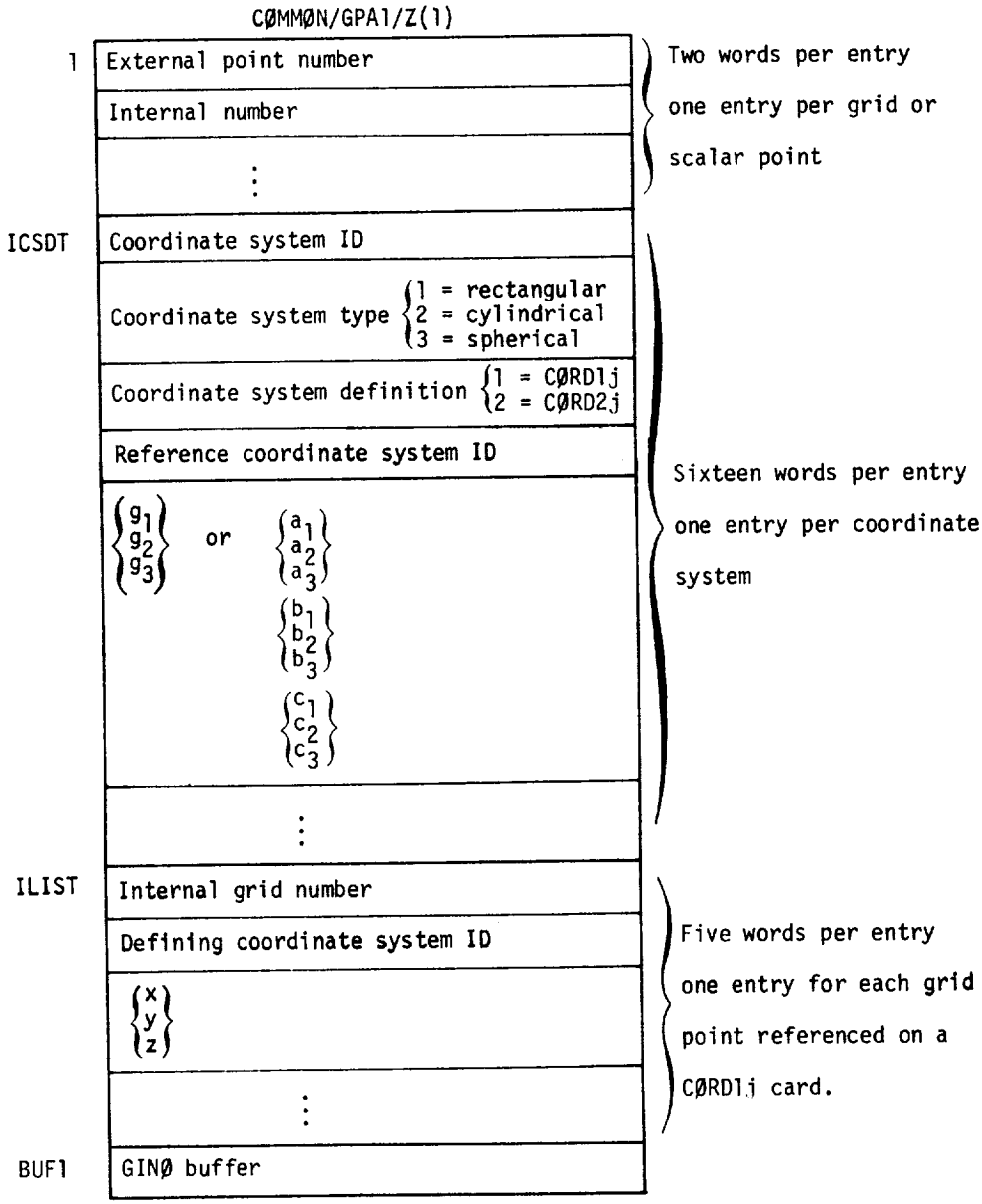
4.21.8 Subroutines

GP1 has no auxiliary subroutines.

4.21.9 Design Requirements

4.21.9.1 Allocation of Core Storage

During the assembly of the GPL, space for 2*(number of grid points plus number of scalar points) plus two GINØ buffers is required. During the assembly of the CSTM, core storage is allocated as follows:



Total storage requirements during this phase, therefore, equals 2*(number of grid + number of scalar points) + 16*(number of coordinate systems) + 5*(number of grid points referenced on CØRD1j cards) + one GINØ buffer.

4.21.9.2 Environment

Open core for GP1 is defined by /GPA1/. The table /GPTA1/ must be in core when GP1 is executed. GP1 uses two scratch files.

4.21.10 Diagnostic Messages

The following diagnostic messages may be issued by GP1:

2001, 2002, 2003, 2004, 2005, 2006, 2012

4.22 FUNCTIONAL MODULE GP2 (GEOMETRY PROCESSOR - PHASE 2)

4.22.1 Entry Point: GP2

4.22.2 Purpose

GP2 processes element connection data and converts external point numbers to internal numbers.

4.22.3 DMAP Calling Sequence

```
GP2 GEØM2,EQEXIN/ECT  $
```

4.22.4 Input Data Blocks

GEØM2 - Element connection data.

EQEXIN - Equivalence between external grid or scalar numbers and internal numbers.

Note: EQEXIN may not be purged.

4.22.5 Output Data Blocks

ECT - Element Connection Table.

Note: ECT may not be purged.

4.22.6 Parameters

None

4.22.7 Method

The first data record of EQEXIN (containing pairs of external point identification and internal index) is read into core. GEØM2 is opened, and the header record is skipped. The ECT is opened, and the header record is written. The following process is repeated for each logical record in GEØM2.

1. The 3-word header is read. If an end-of-file is encountered, step (4) is executed. Otherwise, /GPTA1/ (see description in section 2.5) is searched for a match. If found, step (2) is executed. If not found, an internal table, CARDS, which defines additional cards processed

by GP2 (e.g. GENEL) is searched. If a match is found, step (3) is executed. Otherwise, the record is skipped, and step (1) is repeated.

2. The 3-word header from GEØM2 is written on the ECT. Parameters defining the element are fetched from /GPTA1/. If the number of words per element is less than 5, the sort flag in the GEØM2 trainer is fetched. Each element card of the current type on GEØM2 is read. Each external grid identification is converted to an internal index by performing a binary search in the EQEXIN table. If the point is not in the table, an error message is queued and the NØGØ flag is turned on. If the data is not to be sorted, the element is written directly on the ECT. Otherwise it is saved in core (or written on a scratch file if core is full). When all elements of a given type have been processed, the sort flag is again tested. If off, the ECT record is closed and return to step (1) is made. Otherwise, the data are sorted by SØRT and the ECT record is then written.

3. For GENEL, SEQBFE and QDSEP data (the latter two are Force Method only), each entry is read, all external point identifications are converted to internal indices as in (2) and the entry is written on the ECT. When the logical record on GEØM2 is exhausted, the ECT record is closed and return to step (1) is made.

4. The ECT trailer is written, and all files are closed. If the NØGØ flag was turned on, PEXIT is called. Otherwise, a normal exit is made.

4.22.8 Subroutines

The module GP2 consists of one subroutine, GP2.

4.22.9 Design Requirements

4.22.9.1 Allocation of Core Storage

GP2 requires space for 2*(number of grid points + number of scalar points) + three GINØ buffers.

4.22.9.2 Environment

Open core is defined by /GPA2/. The table /GPTA1/ must be in core when GP2 is executed. GP2 uses up to four scratch files.

4.22.10 Diagnostic Messages

The following diagnostic messages may be issued by GP2:

2007, 2059, 2060, 2061

4.23 FUNCTIONAL MODULE PLTSET (PLOT SET DEFINITION PROCESSOR)

4.23.1 Entry Point: DPLTST

4.23.2 Purpose

To generate the structural element sets to be used by the structural plotter (functional module PLØT).

4.23.3 DMAP Calling Sequence

```
PLTSET     PCDB,EQEXIN,ECT/PLTSETX,PLTPAR,GPSETS,ELSETS/V,N,NGP/V,N,NPSET    $
```

4.23.4 Input Data Blocks

PCDB - Plot Control Data Block for the structure plotter.

EQEXIN - Equivalence between external grid or scalar numbers and internal numbers.

ECT - Element Connection Table.

Note: If PCDB is purged, nothing is done in this module. However, if PCDB is not purged, neither EQEXIN nor ECT may be purged.

4.23.5 Output Data Blocks

PLTSETX - User error messages related to the definition of element plot sets for the structure plotter.

PLTPAR - Plot parameters and plot control table.

GPSETS - Grid point sets related to the element plot sets.

ELSETS - Element plot set connection tables.

Note: None of these data blocks may be pre-purged unless PCDB is also purged.

4.23.6 Parameters

NGP - Output-integer-no default. Total number of grid points.

NPSET - Output-integer-default value = -1. Number of element plot sets (set to -1 if none).

4.23.7 Method

Each logical card in the plot control data block (PCDB) is read. If the first entry on a card is not "SET", the card is assumed to be a plot parameter or control card meaningful only to the PLØT module. In this case, the logical card is copied onto the PLTPAR data block.

If the first entry on a card is "SET", it is assumed to be a definition of a new element plot set. As each entry on the card is read, it is decided whether a list of elements (by type, range or explicit id's) or a list of grid points (by range or explicit id) is being included or excluded. Each element type which is specified is inserted into a table (TYP, 100 words long). If a range of elements or an explicit element id is specified, it is inserted into the beginning of open core (the EL array). And finally, if a range of grid points or an explicit grid point id is specified, it is inserted into the end of open core (the GP array). When a set has been completely specified, it is written out onto a scratch file (MSET) in the following format:

Word	1	=	NEL (number of entries in the EL array).
Word 2	to NEL+1	=	the entries in the EL array.
Word	1	=	NTYP (number of entries in the TYP array).
Word 2	to NTYP+1	=	the entries in the TYP array.
Word	1	=	NPT (number of entries in the GP array).
Word 2	to NPT+1	=	the entries in the GP array.

After all the SET cards have been processed, subroutine CØMECT is called to set up a shortened element connection table (ECTX). For each element type, the table is as follows:

	Word 1	=	two character BCD element type symbol (left justified)
	Word 2	=	number of grid points per element.
for each element of this type	Word 3	=	element id.
	Word 4, etc.	=	internal grid point numbers of the grid points connected by this element

This table, in conjunction with MSET, is used by subroutine CNSTRC to create the GPSETS and ELSETS data blocks. The ELSETS data block is simply a duplicate of ECTX for each plot set, except that only those elements which are in the set are included. The GPSETS data block for each plot set is simply a list of indices into the subset of grid points which pertain to this set.

4.23.8 Subroutines

Utility routines CLSTAB, FREAD, GØPEN, INTGPX, INTGPT, INTLST, RDMØDX, RDMØDE and RDWØRD are used by PLTSET. See section 3.4 for their descriptions.

4.23.8.1 Subroutine Name: SETINP

1. Entry Point: SETINP

2. Purpose: To create the plot parameter and control data block (PLTPAR) and interpret the plot set definition cards from the Plot Control Data Block (PCDB).

3. Calling Sequence: CALL SETINP

CØMMØN/EDES/NTYPES,ENAM(2,50) - see EDESBD Block Data description below (section 4.23.9.2).

4.23.8.2 Subroutine Name: CØMECT

1. Entry Point: CØMECT

2. Purpose: To create a shortened form of the Element Connection Table (ECT).

3. Calling Sequence: CALL CØMECT

CØMMØN/EDES/NTYPES,ENAM(2,50),ECØD(2,50),EWRD(50),ENGP(50) - see EDESBD Block Data description.

4.23.8.3 Subroutine Name: CNSTRC

1. Entry Point: CNSTRC

2. Purpose: To construct the element and grid point plot set data blocks (ELSETS,GPSETS).

3. Calling Sequence: CALL CNSTRC (GP,ELE,BUF,MAX)

CØMMØN/EDES/NTYPES - see EDESBD Block Data description, section 4.23.9.2.

Where:

GP = NGP locations used to set up the grid point index list for the grid point set data block (GPSETS).

ELE = MAX locations used to set up the element set data block (ELSETS).

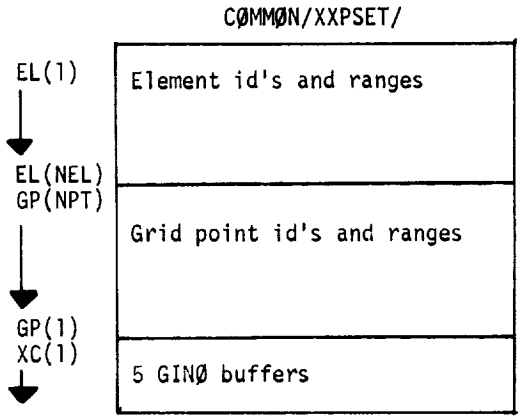
BUF = Location of 3 GINØ buffers.

MAX = Amount of core available for the ELE array (open core).

4.23.9 Design Requirements

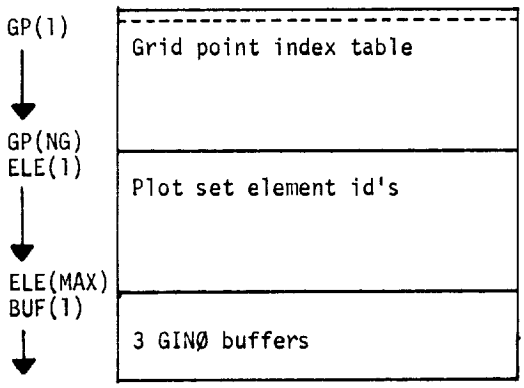
4.23.9.1 Open Core Design (Common Block XXPSET)

1. Subroutine SETINP



2. Subroutine CNSTRC

For this subroutine, open core is partitioned by the calling program as follows:



4.23.9.2 Block Data Interface (EDESBD-Element Description Table)

CØMMØN /EDES/NTYPES,ENAM(2,50),ECØD(2,50),EWRD(50),ENGP(50),ESYM(50)

Where:

NTYPES = number of element types defined in this table.

ENAM = two word BCD name of the element type.

ECØD = two word subroutine LØCATE code of the element type for the ECT data block.

EWRD = number of words per element on the ECT data block.

ENGP = number of grid point connections per element.

ESYM = two character BCD element type symbol (left justified)

Note: The element types defined in this table are the only ones which can be plotted by the structural plotter.

4.23.10 Diagnostic Messages

A fatal message occurs in SETINP if a set specification is so large that open core is filled (i.e., array EL meets array GP). All other diagnostics are non-fatal and are written on the PLTSETX data block for printing by the PRTMSG module. These messages are all quite self-explanatory and straightforward.

4.24 FUNCTIONAL MODULE PLØT (STRUCTURAL PLOTTER)

4.24.1 Entry Point: DPLØT

4.24.2 Purpose

To draw structural shapes on a variety of different plotters.

4.24.3 DMAP Calling Sequence

```
PLØT    PLTPAR,GPSETS,ELSETS,CASECC,BGPDT,EQEXIN,SIL,PLTDSP1,PLTDSP2/PLØTX/V,N,NGP/
        V,N,LSIL/V,N,NPSET/V,N,PLTFLG/V,N,PLTNUM    $
```

4.24.4 Input Data Blocks

PLTPAR - Plot parameters and plot control table.

GPSETS - Grid point sets related to the element plot sets.

ELSETS - Element plot set connection tables.

CASECC - Case Control Data Table.

BGPDT - Basic Grid Point Definition Table.

EQEXIN - Equivalence between external grid or scalar numbers and internal numbers.

SIL - Scalar Index List.

PLTDSP1 - Translational displacements (statics).

PLTDSP2 - Translational displacements (dynamics).

Notes:

1. Only SIL, PLTDSP1, and PLTDSP2 may be purged. If this is the case, only undeformed shapes may be drawn.

2. If either PLTSDP1 or PLTDSP2 is purged, that type of deformed shape will not be drawn.

3. If either PLTDSP1 or PLTDSP2 is not purged, SIL may not be purged.

4.24.5 Output Data Blocks

PLØTX - User messages.

Note: PLØTX may not be purged.

4.24.6 Parameters

NGP - Integer-input-no default value. Number of grid points.

LSIL - Integer-input-no default value. Last scalar index value.

NPSET - Integer-input-no default value. Number of element plot sets.

PLTFLG - Integer-input/output-default value = 1. Displacement plot flag.

= 1 if undeformed shapes have not yet been drawn.

= -1 if undeformed shapes have been drawn.

PLTNUM - Integer-input/output-default value = 0. Plot number.

4.24.7 Method

Subroutine PARAM reads each card in the plot parameter and control table (PLTPAR). If the first entry on a card is not 'FIND' or 'PLØT', it is assumed to be a plot parameter card to be processed within PARAM (e.g., PRØJECTIØN, PLØTTER, etc.). Within PARAM, an implied 'FIND' card is initially set up to automatically find an origin, vantage point, and scale. In addition this same implied "FIND" card is set up each time a new projection is defined as a 'PLØTTER' card is encountered. At the same time, the view angles are re-initialized to their default values, the regions pertaining to each origin are reset to full pictures, and all previously defined origins are nullified.

When a 'FIND' card is encountered, subroutine FIND is called both to interpret the card and act upon its requests. And finally, when a 'PLØT' card is encountered, subroutine PLØT is called both to interpret the card and to act upon its requests. However, in this case, if the implied 'FIND' card set up by subroutine PARAM still exists (i.e., if no origin, scale, or vantage point has been defined) the FIND subroutine is called to satisfy these needs before subroutine PLØT is called.

In subroutine FIND, after the interpretation of the 'FIND' card is completed, a coordinate system rotation matrix is calculated relative to the current view angles, and then the vantage point, scale factors, and the origin requested are calculated as needed.

In subroutine PLØT, after the interpretation of the 'PLØT' card is completed, a list of messages to the plotter operator is generated. Then all plots requested on the plot card are generated by calling subroutine DRAW for each plot request.

Subroutine DRAW generates one plot. It sets up the region of the plot, rotates the grid points based upon the current viewing angles, applies the latest scale factor to the structural coordinates, and translates these coordinates to the origin specified for this plot. It also controls the various aspects of a plot as specified on the 'PLØT' card, e.g., drawing a shape, labeling grid points, etc., for both undeformed and deformed structures (superimposition if called for).

4.24.8 Subroutines

The following utility routines are called by PLØT: CLSTAB,FREAD,GØPEN,(INTGPX,INTGPT), INTLST,(RDMØDX,RDMØDY,RDMØDE,RDWØRD). See the subroutine descriptions in section 3. The subroutines FNDSET, MINMAX, PERPEC and PROCES are support subroutines used by more than one of the following subroutines.

4.24.8.1 Subroutine Name: PARAM

1. Entry Point: PARAM

2. Purpose: To interpret the plot parameter cards and to detect the 'FIND' and 'PLØT' plot control cards. In addition, it serves as a driver for subroutine FIND and PLØT.

3. Calling Sequence: CALL PARAM (SETID,X)

CØMMØN/XXPARM/ - See XXPARM table description below (section 4.24.9.2).

CØMMØN/PLTDAT/ - See PLTDAT miscellaneous table description (section 2.5).

Where:

SETID = Various plot set id's created in the PLTSET module. (Record 1 of GPSETS).

X = Open core.

4. Method: All plot parameters are inserted in the XXPARM table. Any parameter which is not recognizable causes a message to be created to this effect, and the parameter is then ignored.

5. Additional Subroutines Required: FIND,PLØT

4.24.8.2 Subroutine Name: FIND

1. Entry Point: FIND

2. Purpose: To interpret a 'FIND' card and to calculate the parameters requested on the card.

3. Calling Sequence: CALL FIND (MØDE,SETID,X)

CØMMØN/XXPARM/ - See XXPARM table description below (section 4.24.9.2).

CØMMØN/PLTDAT/ - See PLTDAT miscellaneous table description (section 2.5).

Where:

MØDE = Current value of the XRCARD mode value as read and modified by subroutine

RDMØDX.

SETID = Various plot set id's created in the PLTSET module (record 1 of GPSETS).

X = Open core.

4. Method: After interpreting the 'FIND' card, the coordinate system rotation matrix is calculated (based upon the current view angles), the vantage point, scale factor, and desired origin are calculated.

5. Additional Subroutines Required: FNDSET,MINMAX,PRØCES,PERPEC.

4.24.8.3 Subroutine Name: PLØT

1. Entry Point: PLØT

2. Purpose: To interpret the 'PLØT' card, and produce all the plots requested on the card by acting as a driver to subroutine DRAW.

3. Calling Sequence: CALL PLØT (MØDE,SETID,X)

CØMMØN/XXPARM/ - See XXPARM table description below (section 4.24.9.2).

CØMMØN/PLTDAT/ - See PLTDAT miscellaneous table description (section 2.5).

Where:

MØDE = Current value of the XRCARD mode value as read and modified by subroutine RDMØDX.

SETID = Various plot set id's created in the PLTSET module (record 1 of GPSETS).

X = Open core.

4. Method: After interpreting the deformed structure plot requests (there may be many on one 'PLØT' card), the rest of the 'PLØT' card is read into memory. For each deformed structure request, the appropriate displacement data block (PLTDSP1 or PLTDSP2) is searched for a matching subcase id. If one is found, (this search does not occur if only the undeformed requests are being serviced), then the rest of the plot card is interpreted for the various plotting options. Subroutine DRAW is then called to service these options and to draw the corresponding picture for each plot element set listed on the 'PLØT' card.

5. Additional Subroutines Required: HEAD,FNDSET,GETDEF,DRAW.

4.24.8.4 Subroutine Name: GETDEF

1. Entry Point: GETDEF

2. Purpose: To read the translational components of a set of displacements (in the basic coordinate system).

3. Calling Sequence: CALL GETDEF (DFRM,GPT,D)

CØMMØN/XXPARM/ - See XXPARM table description below (section 4.24.9.2).

where:

DFRM = Displacement data block to be read (pre-positioned at the set of displacements to be read).

GPT = List of grid point indices defining a subset of grid points.

D = Array into which the displacement components are to be read (3 per grid point - X,Y,Z).

4. Method: The scalar index list (SIL data block) is used to determine at which grid point a particular displacement component is specified. While reading the components, a maximum absolute component (MAXDEF) is determined.

4.24.8.5 Subroutine Name: PLTØPR

1. Entry Point: PLTØPR

2. Purpose: To generate printed output to be used by the plotter operator in setting up the plotting equipment, and to generate output informing the user of the plotting parameters used to generate the plots.

3. Calling Sequence: CALL PLTØPR

CØMMØN/PLTDAT/ - See PLTDAT miscellaneous table description (section 2.5).

CØMMØN/XXPARM/ - See XXPARM table description below (section 4.24.9.2).

4. Method: All output is written on the PLØTX data block for subsequent processing by the PRTMSG module. The resulting output can be used by the user to alter certain plot parameters on a subsequent run, if he desires, in order to slightly alter the plots produced.

4.24.8.6 Subroutine Name: DRAW

1. Entry Point: DRAW

2. Purpose: To service the many possible plotting options and generate the corresponding picture.

3. Calling Sequence: CALL DRAW (GPLST,X,U,S,DEFØRM,STEREØ)

CØMMØN/PLTDAT/ - See PLTDAT miscellaneous table description (section 2.5).

CØMMØN/XXPARM/ - See XXPARM table description below (section 4.24.9.2).

CØMMØN/RSTXXX/ - See RSTXXX table description below (section 4.24.9.2).

CØMMØN/DRWDAT/ - See DRWDAT table description below (section 4.24.9.2).

where:

GPLST = List of indices (one for each structural grid point) into the subset of grid points which pertain to the element set appropriate to this plot.

X = Coordinates of the grid points in this element set (3 per grid point - r,s,t).

U = Deformation components for each grid point in this element set (3 per grid point - x,y,z).

S = Location into which the s and t deformed structure grid point coordinates are to be placed.

DEFØRM { = 0 if an undeformed structure plot is requested.
= 1 if a deformed structure plot is requested.

STEREØ { = 0 if the left image of a stereo plot is to be generated.
= 1 if the right image of a stereo plot is to be generated.

4. Method: Initially, the grid points are rotated based upon the current viewing angles, translated to the selected plot origin, and converted to plotter units using the current scale factor, and the deformation components are reduced to the specified maximum deformation value. Then the undeformed structural plot is generated.

Next, the deformed structural shape (if requested) is drawn. Then the deformation vectors (as requested) are drawn.

5. Additional Subroutines Required: MINMAX,PRØCES,PERPEC,INTVEC,SHAPE,GPTSYM,GPTLBL,DVECTR, ELELBL

4.24.8.7 Subroutine Name: INTVEC

1. Entry Point: INTVEC

2. Purpose: To interpret the user supplied deformation vector plot request.

3. Calling Sequence: CALL INTVEC (VECTØR)

where:

VECTØR = BCD characters specified in the deformation vector request (any combination of the characters R, X, Y, Z). Input and Output. On input, VECTØR is integer (=0) or BCD. On output, VECTØR is integer (=0, if = 0 upon input).

4. Method: The result is stored into VECTØR, as follows:

$X = 2^0$

$Y = 2^1$

$Z = 2^2$

$R = 2^3$ (if VECTØR = 'R' only, it is treated as if VECTØR = RXYZ).

4.24.8.8 Subroutine Name: SHAPE

1. Entry Point: SHAPE

2. Purpose: To draw a structural shape.

3. Calling Sequence: CALL SHAPE (GPLST,X,U,PEN,DEFØRM)

where:

GPLST = List of indices into the subset of grid points pertaining to the shape to be drawn.

X = Corresponding grid point coordinates of the undeformed structure (3 per grid point - r, s, t).

U = Corresponding grid point coordinates of the deformed structure (2 per grid point - s, t).

PEN = Pen number or line density to be used to draw the shape.

DEFØRM { = 0 if the undeformed shape is to be drawn.
DEFØRM { = 1 if the deformed shape is to be drawn.

4. Method: The structural shape to be drawn is defined as a compact element connection table on the ELSETS data block (assumed open and positioned at the correct element set). As each element is read, it is drawn, taking into account whether the element is one or two dimensional.

4.24.8.9 Subroutine Name: GPTSYM

1. Entry Point: GPTSYM

2. Purpose: To type a special symbol at each of a subset of grid points.

3. Calling Sequence: CALL GPTSYM (GPLST,X,U,SYM,DEFØRM)

Where:

GPLST = List of indices defining the subset of grid points.

X = Corresponding grid point coordinates of the undeformed structure (3 per grid point - x,s,t).

U = Corresponding grid point coordinates of the deformed structure (2 per grid point - s,t).

SYM = Two indices to be used to construct the special symbol.

DEFØRM { = 0 if the undeformed grid points are to be used.
DEFØRM { = 1 if the deformed grid points are to be used.

4.24.8.10 Subroutine Name: GPTLBL

1. Entry Point: GPTLBL

2. Purpose: To type the external grid point id of each of a subset of grid points.

3. Calling Sequence: CALL GPTLBL (GPLST,X,U,DEFØRM)

CØMMØN/PLTDAT/ - See the PLTDAT miscellaneous table description (section 2.5).

Where:

GPLST = List of indices defining the subset of grid points.

X = Corresponding grid point coordinates of the undeformed structure (3 per

grid point - r,s,t).

U = Corresponding grid point coordinates of the deformed structure (2 per grid point - s,t).

DEFØRM { = 0 if the undeformed grid points are to be used.
1 if the deformed grid points are to be used.

4. Method: The internal and external id of each structural grid point is read from the EQEXIN data block. If the grid point is part of the specified subset, then the external id is printed to the immediate right of the grid point.

4.24.8.11 Subroutine Name: DVECTR

1. Entry Point: DVECTR

2. Purpose: To draw deformation vectors.

3. Calling Sequence: CALL DVECTR (GPT,X,U,PEN)

Where:

GPT = List of indices defining the subset of grid points at which vectors are to be drawn.

X = Corresponding grid point coordinates of the undeformed structure (3 per grid point - x,s,t).

U = Corresponding grid point coordinates of the deformed structure (2 per grid point - s,t).

PEN = Pen number or line density to be used to draw the vectors.

4.24.8.12 Subroutine Name: FNDSET

1. Entry Point: FNDSET

2. Purpose: To find the subset of grid points pertaining to a set of elements, and to read the corresponding grid point coordinates from the BGPDT data block.

3. Calling Sequence: CALL FNDSET (SET,GPID,X)

Where:

SET = Element plot set index.

GPID - Array into which the list of indices defining the subset of grid points

is to be read.

X = Array into which the corresponding coordinates are to be read (3 per grid point - x,y,z).

4. Method: If SET = 0, the grid point set index data block (GPSETS) is assumed positioned at the correct record. Otherwise, GPSETS is first positioned correctly (record SET+1). The indices are then read into GPID, and the corresponding coordinates are read from BGPDT into X.

4.24.8.13 Subroutine Name: MINMAX

1. Entry Point: MINMAX

2. Purpose: To initialize the minimum and maximum grid point coordinates to a very large and small number, respectively.

3. Calling Sequence: CALL MINMAX

CØMMØN/RSTXXX/ - See the RSTXXX table description below (section 4.24.9.2).

4.24.8.14 Subroutine Name: PERPEC

1. Entry Point: PERPEC

2. Purpose: To calculate the vantage point and/or translate the grid point coordinates to the vantage point.

3. Calling Sequence: CALL PERPEC (X,STEREØ)

CØMMØN/XXPARM/ - See XXPARM table description (section 4.24.9.2).

CØMMØN/RSTXXX/ - See the RSTXXX table description (section 4.24.9.2).

where:

X = = Set of grid point coordinates to be translated (3 per grid point - r,s,t)

STEREØ = {0 if the coordinates of the left image for stereo are to be calculated.
{1 if the coordinates of the right image for stereo are to be calculated.

4. Method: After the vantage point is calculated (if required), each grid point is translated. In the process, unless the projection is stereo, the minimum and maximum s and t coordinates are calculated. Finally, the differences between these minima and

maxima, and their averages, are calculated.

4.24.8.15 Subroutine Name: PRØCES

1. Entry Point: PRØCES

2. Purpose: To exchange coordinate axes (as requested) and rotate the grid point coordinates based upon the current view angles.

3. Calling Sequence: CALL PRØCES (X)

CØMMØN/XXPARM/ - See the XXPARM table description below (section 4.24.9.2).

CØMMØN/RSTXXX/ - See the RSTXXX table description below (section 4.24.9.2).

where:

X = Grid point coordinates (3 per grid point - x,y,z).

4. Method: In addition to its primary purpose, this subroutine calculates the minimum and maximum rotated grid point coordinates, and the differences and averages of these minima and maxima.

4.24.8.16 Subroutine Name: ELELBL

1. Entry Point: ELELBL

2. Purpose: To type the element identification number of each element in a subset of elements.

3. Calling Sequence: CALL ELELBL (GPLST,X,U,DEFØRM)

CØMMØN/CHAR94/ - See the CHAR94 miscellaneous table description (section 2.5).

CØMMØN/PLTDAT/ - See the PLTDAT miscellaneous table description (section 2.5).

where:

GPLST = List of indices defining the set of grid points associated with the elements to be labeled.

X = Corresponding grid point coordinates of the undeformed structure.

U = Corresponding grid point coordinates of the deformed structure.

$$\text{DEFØRM} = \begin{cases} 0 \text{ if the undeformed grid points are to be used} \\ 1 \text{ if the deformed grid points are to be used} \end{cases}$$

4. Method: The compact element connection table (ELSETS) is read. As each element id is read, it is typed at the center of the element. The two character symbolic name of the element type is appended to the element id.

4.24.9 Design Requirements

4.24.9.1 Open Core Design (Common Block XXPLØT)

Define NPSET = Number of element plot sets.

NGP = Total number of grid points.

NGPSET = Number of grid points in an element.

1. Subroutine DPLØT partitions open core for subroutine PARAM as follows:

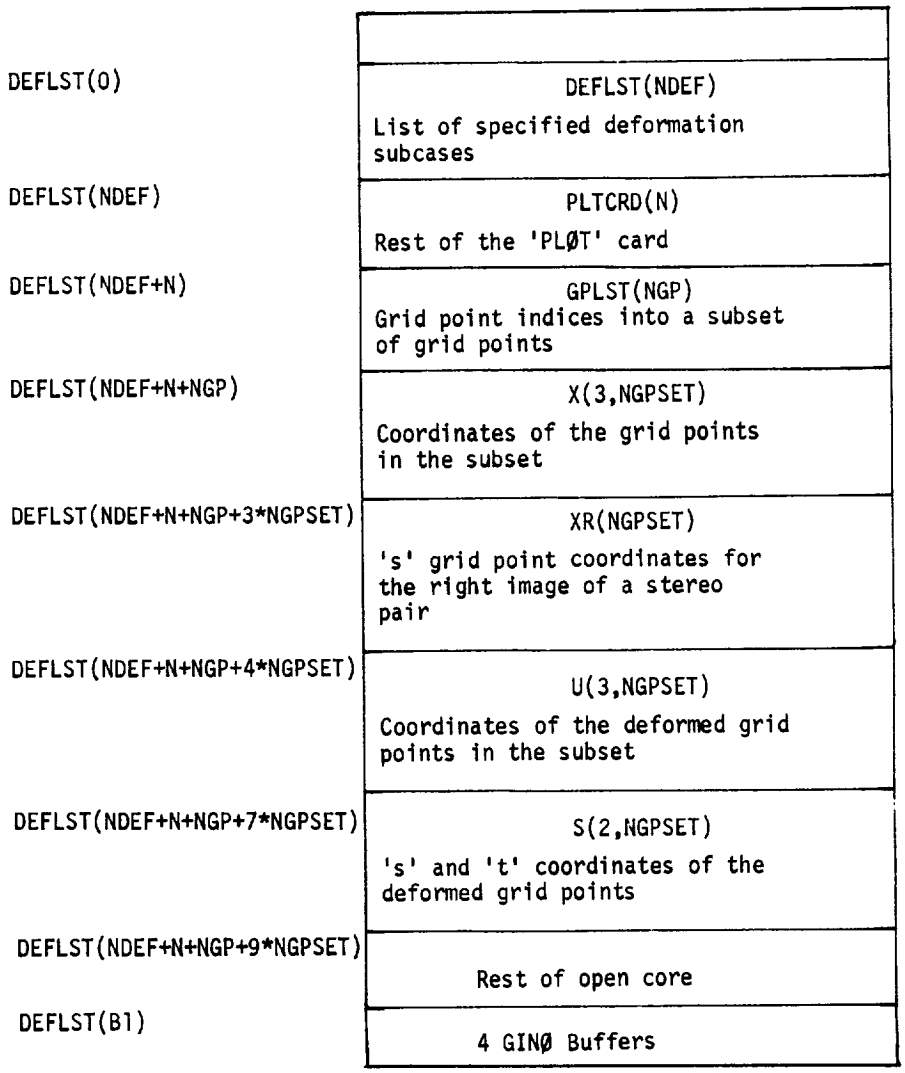
CØMMØN/XXPLØT/

X(0)	Element plot set id's
X(NPSET)	Open Core
X(BUF)	3 GINØ buffers

2. Subroutine FIND partitions open core for subroutines FNDSET, PERPEC, and PRØCES as follows:

X(0)	GPLST(NGP) Grid point indices into a subset of grid points
X(NGP)	X(3,NGPSET) Coordinates of the grid points in the subset
X(NGP+3*NGPSET)	XR(NGPSET) 's' grid point coordinates for the right image of a stereo pair.
X(NGP+4*NGPSET)	Rest of open core

3. Subroutine PLØT partitions open core for subroutines FNDSET, GETDEF, and DRAW as follows:



4.24.9.2 Block Data Interface

1. CØMMØN/DRWDAT/ SET,LABEL,ØRIGIN,PEN,SHAPE,SYMBØL(2),SYM(6),VECTØR

SET	-	Element plot set index.	Integer
LABEL	-	Grid point label option.	
ØRIGIN	-	Origin index.	
PEN	-	Pen number or density value.	
SHAPE	-	Structural shape drawing option.	
SYMBØL	-	Grid point symbol indices.	
SYM	-	Symmetry options.	
VECTØR	-	Deformation vector options.	

2. CØMMØN/RSTXXX/ CSTM(3,3),MIN(3),MAX(3),D(3),AVER(3)

CSTM	-	3x3 coordinate system rotation matrix.	Real
MIN	-	Minimum rotated grid point coordinates.	
MAX	-	Maximum rotated grid point coordinates.	
D	-	Differences between the minima and maxima.	
AVER	-	Averages of the minima and maxima.	

3. CØMMØN/XXPARM/ PBUFSZ,CAMERA,BFRAMS,PLTMDL(2),TAPDEN,
NPENS,PAPSIZ(2),PAPTYP(2),PENSIZ(8),PENCLR(8,2),"SKIP(1)",
SCALE,ØBJMØD,FSCALE,MAXDEF,DEFMAX,
AXIS(3),DAXIS(3),VANGLE(3),BETAØS,BETAP,"SKIP(4)",
FVP,RO,SOL,SOR,TO,DO,DO2,DO3,PRJECT,SOS
FØRG,ØRG,NØRG,ØRIGIN(11),EDGE(11,4),XY(11,3)

In the following descriptions, the value(s) in parentheses to the right of the variable name, the default value, and the letter in parentheses to the right of the explanation pertain to the type of the variable (I implies integer and R implies real).

PBUFSZ(0) = Plot tape buffer size (I)

Plotter Data

CAMERA(2)	= Plotter camera number (I).
BFRAMS(1)	= Number of blank frames between plots (I).
PLTMDL(4020,0)	= Plotter model (BCD or I or R).
TAPDEN(0)	= Plot tape density (I).

Pen and Paper Data

NPENS(8)	= Maximum number of pens (I).
PAPSIZ(8.5,11.)	= Paper size in inches (R).
PAPTYP(VELLUMbb)	= Paper type (BCD).
$PENSIZ_i$(1)	= Pen sizes (I).
$PENCLR_{i,1}$(BLAC) & $PENCLR_{i,2}$(Kbbb)	= Pen colors (BCD).

Scaling Data

SCALE	= Object-to-plotter or model-to-plotter (stereo only) scale factor (R).
ØBJMØD(1.)	= Object-to-model scale factor (R-stereo only).
FSCALE(1)	= Find scale factors option (I).
MAXDEF(0.)	= Forced value of the largest deformation component (R).
DEFMAX	= Actual largest deformation component (R).

Viewing Data

AXIS (1,2,3)	= Undeformed structure axis orientation (I).
DAXIS(1,2,3)	= Deformed structure axis orientation (I).
VANGLE(0.,-1.10^{10}, 34.27)	= View angles (R-alpha,beta,gamma).
BETAØS(23.17)	= Orthographic and stereo default value for the "beta" view angle (R).
BETAP(0.)	= Perspective default value for the "beta" view angle (R).

Projection Data

FVP(1)	= Find vantage point option (I).
RO	= "r" component of the vantage point (R).
SOL	= "s" component of the perspective vantage point (R).
SOL,SOR	= "s" components of the stereo vantage point (R).
TO	= "t" component of the vantage point (R).
DO	= Projection plane separation value (R).
DO2(1.)	= Perspective default projection plane separation value (R).
DO3(2.)	= Stereo default projection plane separation value (R).
PRJECT(1)	= Projection type (I, 1=orthographic, 2 = perspective, 3 = stereo).
SOS(2.756)	= Ocular separation value (R).

Origin Data

FØRG(1)	= Find origin point option (I).
ØRG(0)	= Number of active origins (I).
NØRG(10)	= Maximum number of active origins (I).
ØRIGIN	= Active origin id's (I).
$EDGE_{i,1}(0.)$ & $EDGE_{i,2}(0.)$	= Lower left corner of the region specified for the ith origin (R).
$EDGE_{i,3}(1.)$ & $EDGE_{i,4}(1.)$	= Upper right corner of the region specified for the ith origin (R).
$XY_{i,1}$	= x component of the i^{th} origin (R).
$XY_{i,3}$	= y component of the i^{th} origin (R).
$XY_{i,1}$ & $XY_{i,2}$	= left and right x components of the i^{th} origin for stereo projection (R).

4.24.9.3 Common Storage Requirements

1. /XXPLØT/ - Open core.
2. /PLTDAT/ - Plotter data (see miscellaneous table description - section 2.5).
3. /XXPARM/ - Plotting parameters.
4. /RSTXXX/ - Plot co-ordinate system calculations.
5. /DRWDAT/ - Drawing data.

4.24.10 Diagnostic Messages

A non-fatal message, number 3008, CALL MESAGE (8,x,x), will be generated by subroutine PLØT if not enough core is available for the grid point data needed for a specific element plot set. If this occurs, this set will not be used to generate a plot.

All other diagnostics are non-fatal and are written onto the PLØTX message data block for printing by the PRTMSG module. These messages are all quite self-explanatory and straightforward, and do not have any external message numbers.

4.25 FUNCTIONAL MODULE GP3 (GEOMETRY PROCESSOR - PHASE 3)

4.25.1 Entry Point: GP3

4.25.2 Purpose

GP3 processes static loads and temperature data. Static load data are collected by load set, and external numbers are converted to internal numbers. Similarly, temperature data are collected by temperature set and external numbers are converted to internal numbers.

4.25.3 DMAP Calling Sequence

```
GP3     GEØM3,EQEXIN,GEØM2/SLT,GPTT/V,N,NØLØAD/V,N,NØGRAV/V,N,NØTEMP  $
```

4.25.4 Input Data Blocks

GEØM3 - Static loads and temperature data.

EQEXIN - Equivalence between external grid and scalar numbers and internal numbers.

GEØM2 - Element connection data.

Note: EQEXIN may not be purged.

4.25.5 Output Data Blocks

SLT - Static Loads Table.

GPTT - Grid Point Temperature Table.

4.25.6 Parameters

NØLØAD - Output-integer-no default. -1 if no static loads (i.e. SLT is not created), +1 otherwise.

NØGRAV - Output-integer-no default. -1 if no GRAV cards in the Bulk Data Deck, +1 otherwise.

NØTEMP - Output-integer-no default. -1 if no TEMP or TEMPD cards in Bulk Data Deck (or if GPTT is purged), +1 otherwise.

4.25.7 Method

Subroutine GP3 is the control program for the module. It executes each of the major subroutines of GP3 (GP3C, GP3A, GP3B) depending on the status of the data blocks. A flow chart for GP3 is included in Figure 1.

4.25.8 Subroutines

4.25.8.1 Subroutine Name: GP3

1. Entry Point: GP3

2. Purpose: Module control program.

3. Calling Sequence: CALL GP3

4.25.8.2 Subroutine: GP3C

1. Entry Point: GP3C

2. Purpose: To convert PLØAD2 data to PLØAD format, merge PLØAD2 data with PLØAD data (if present) and write the resulting data on SCR2, a scratch file.

3. Calling Sequence: CALL GP3C

4. Method: PLØAD2 cards are read into core from GEØM3. Six words are used for each entry. The first word (set identification) is set negative and the sixth word of each entry is set to zero. GEØM2 is opened and the header record is skipped. The following steps occur for each record on GEØM2.

1. The 3-word header is read. /GPTA1/ (see section 2.5) is searched for a match. If no match is found, the record is skipped and the process is repeated. If an end-of-file is encountered, step (3) is executed. If a match is found, a test on element type is made. If a one-dimensional element, the record is skipped and the process repeated. Otherwise, step (2) is executed.

2. An entry of the current element type is read. A linear search through the PLØAD2 data in core is made to find a match on element identification (3rd word of each PLØAD2 entry). If no match is found, the next entry is read. For each match which is found, the grid identification numbers which connect the element are stored

in the corresponding PLØAD2 entry and the first word of the PLØAD2 entry is set positive. When all data for the current element type has been read, a return to step (1) is made.

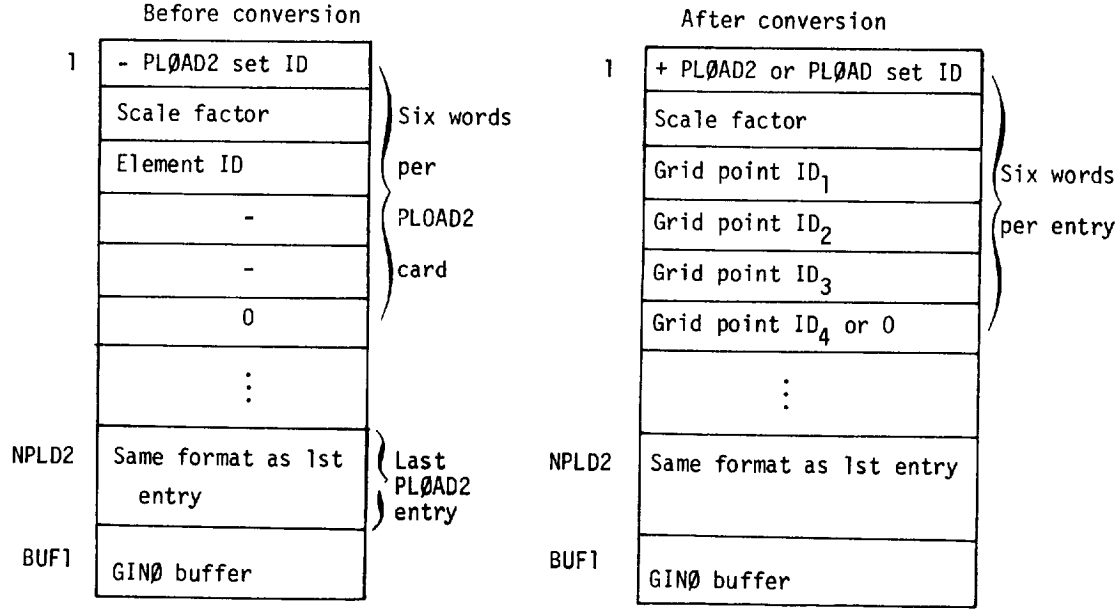
3. A pass through each entry in the PLØAD2 data is made. For each entry for which the first word is negative, an error message is queued and the NØGØ flag turned on. Upon completion of the pass, PEXIT is called if the NØGØ flag was turned on. Otherwise step (4) is executed.

4. LØCATE is called to position GEØM2 to PLØAD data. If none exists, step (5) is executed. Otherwise, the PLØAD data is read into core following the PLØAD2 data. The combined list is sorted by SØRT on set identification number.

5. The data in core is written as one logical record on SCR2. A return to GP3 is given.

Allocation of core storage in GP3C is as follows:

CØMMØN/GP3CØR/Z(1)



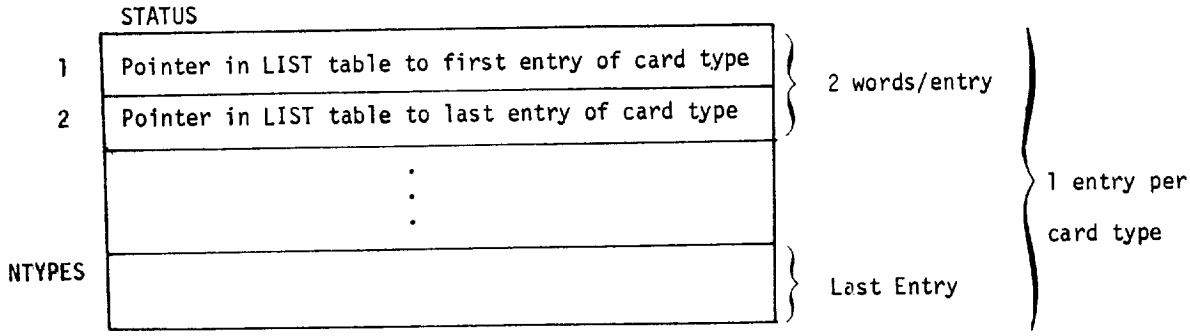
4.25.8.3 Subroutine: GP3A

1. Entry Point: GP3A

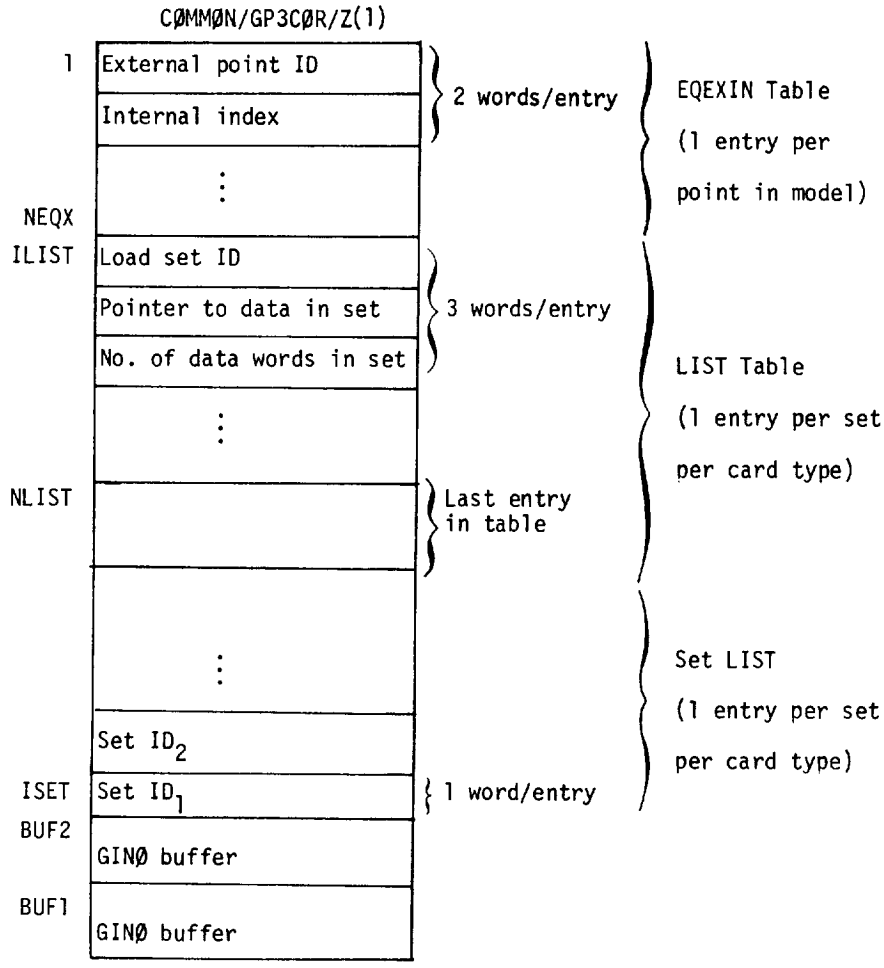
2. Purpose: To assemble to Static Loads Table (SLT).

3. Calling Sequence: CALL GP3A

4. Method: GP3A assembles the SLT by making two passes on the load cards (FØRCEi, MØMENTi, etc). On the first pass each of the cards is read from GEØM3, (or SCR2 for PLØAD data), unique set identifications are extracted and saved in core, all external point identifcations are converted to internal indices by performing a binary search in the EQEXIN table, the data are written on SCR1, and pointer tables are accumulated. These tables are as follows:



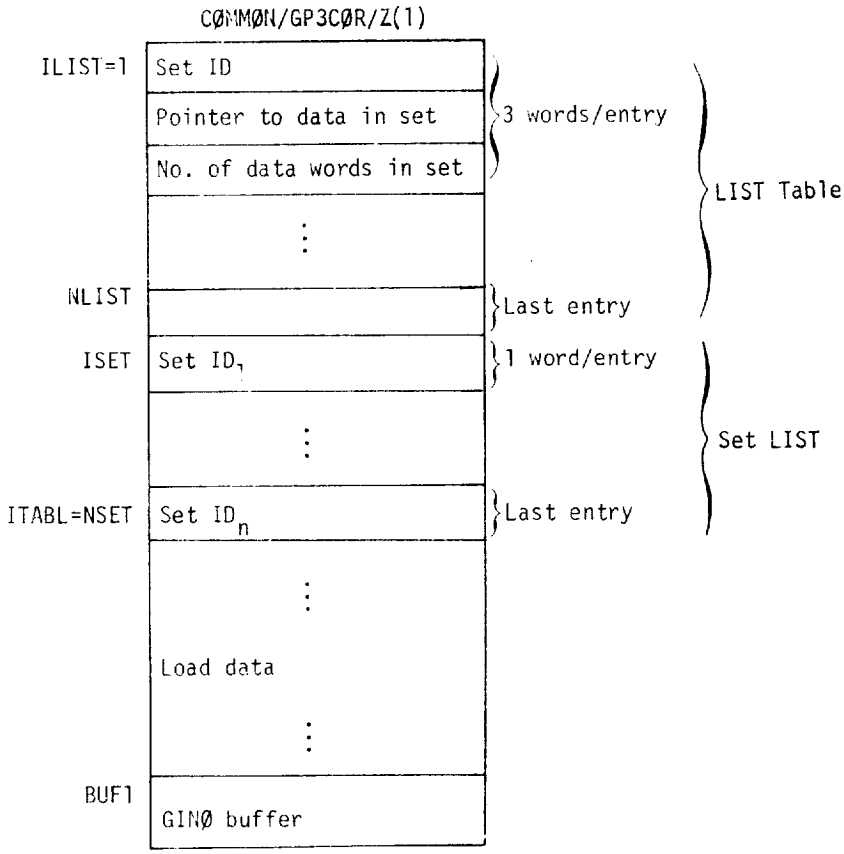
Note: entry = (-1, -1) if card type not present



Note:

Set IDs are stored backward in core. ISET points to first entry, ISET-1 to record entry, etc.

At the end of the first pass, the LIST table is moved to the beginning of open core. The set list is sorted, and duplicate set identifications are discarded. The resulting list is stored immediately following the LIST table. If all data for the load cards will fit in the remaining core, this data is read from SCR1. Core storage is as follows:



The SET list is written in the header record on the SLT. For each set ID in the SET list, the LIST table is searched for a match. When found, the pointer to the data is fetched. The data are sorted on the applied point (except GRAV and PLØAD data) and the data written on the SLT. As a result, each logical record of the SLT contains all data for one set. Finally, if combination load cards are present, they are copied from GEØM3 to the last record of the SLT.

If core will not hold the entire load data, the logic is similar to above except that SCR1 is passed once for each set and only data belonging to a single card type within a set are read into core.

4.25.8.4 Subroutine Name: GP3B

1. Entry Point: GP3B

2. Purpose: To assemble the Grid Point Temperature Table (GPTT).

3. Calling Sequence: CALL GP3B

4. Method: EQEXIN is read into core. A list of default temperatures (TEMPD cards if present) is read from GEØM3. The temperature data (TEMP cards) are read to determine the number of temperature sets, the set identifications and the number of entries in each temperature set. For each temperature set, a three-word entry is written in the header record of the GPTT:

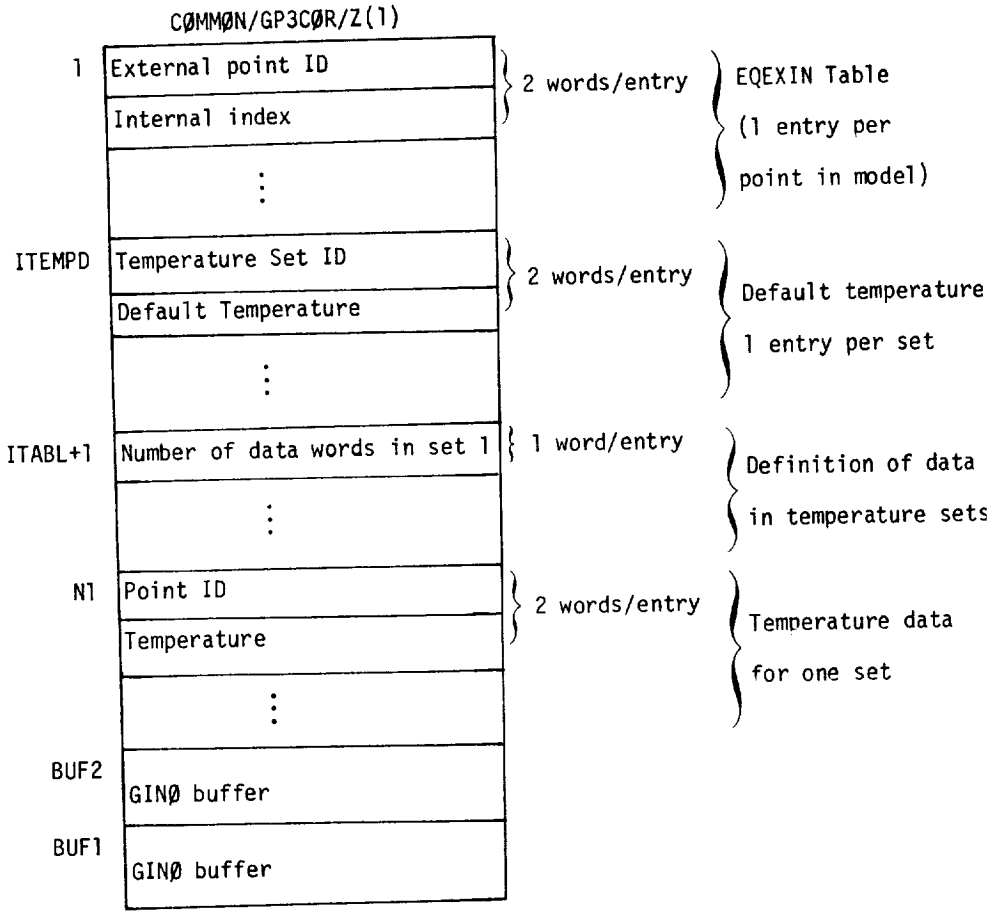
   Word 1 = Set ID

   Word 2 = default temperature (real) or -1 (integer)

   Word 3 = record number in GPTT of temperature data for the set or zero if only default temperature is defined.

GEØM3 is backspaced one logical record. The temperature data are re-read. Each point identification is converted to an internal index by performing a binary search in the EQEXIN table. When all temperature data for a set have been read into core. the data are sorted on internal index and written as one logical record on the GPTT. This process is repeated for each temperature set.

Allocation of core storage for GP3C is as follows:


CØMMØN/GP3CØR/Z(1)
1
External point ID
Internal index
2 words/entry
EQEXIN Table
(1 entry per
point in model)
ITEMPD
Temperature Set ID
Default Temperature
2 words/entry
Default temperature
1 entry per set
ITABL+1
Number of data words in set 1
1 word/entry
Definition of data
in temperature sets
N1
Point ID
Temperature
2 words/entry
Temperature data
for one set
BUF2
GINØ buffer
BUF1
GINØ buffer

4.25.9 Design Requirements

4.25.9.1 Allocation of Core Storage

The core storage maps presented in the method sections of GP3A, GP3B and GP3C provide detailed storage requirements. A summary is presented here.

GP3C: Maximum requirement = 6* (number of PLØAD2 + number of PLØAD cards) + one GINØ buffer.

GP3A: Let NPTS = number of grid + number of scalar points and NSETS = (number of load sets) * (number of card types per load set) and SYSBUF = one GINØ buffer.

Then maximum storage requirement equals MAX ((2*NPTS+4*NSETS+2*SYSBUF), (4*NSETS + MAX (number of words for one set of one card type) + SYSBUF)).

GP3B: See storage map.

4.25.9.2 Environment

1. Block Data

The block data program GP3BD initializes /GP3CØM/ with GINØ file names, data defining the load cards and other miscellaneous data. It must be resident in core when GP3 is executed.

2. General

/GPTA1/ is used by GP3C and must be core resident when GP3 is executed. Open core is defined by /GP3CØR/. The normal overlay is to include GP3BD, GP3, GP3C, GP3A, GP3B in one segment. GP3 uses two scratch files.

4.25.10 Diagnostic Messages

The following messages may be issued by GP3:

2008, 2009, 2015, 3008. See section 6 of the User's Manual for details.

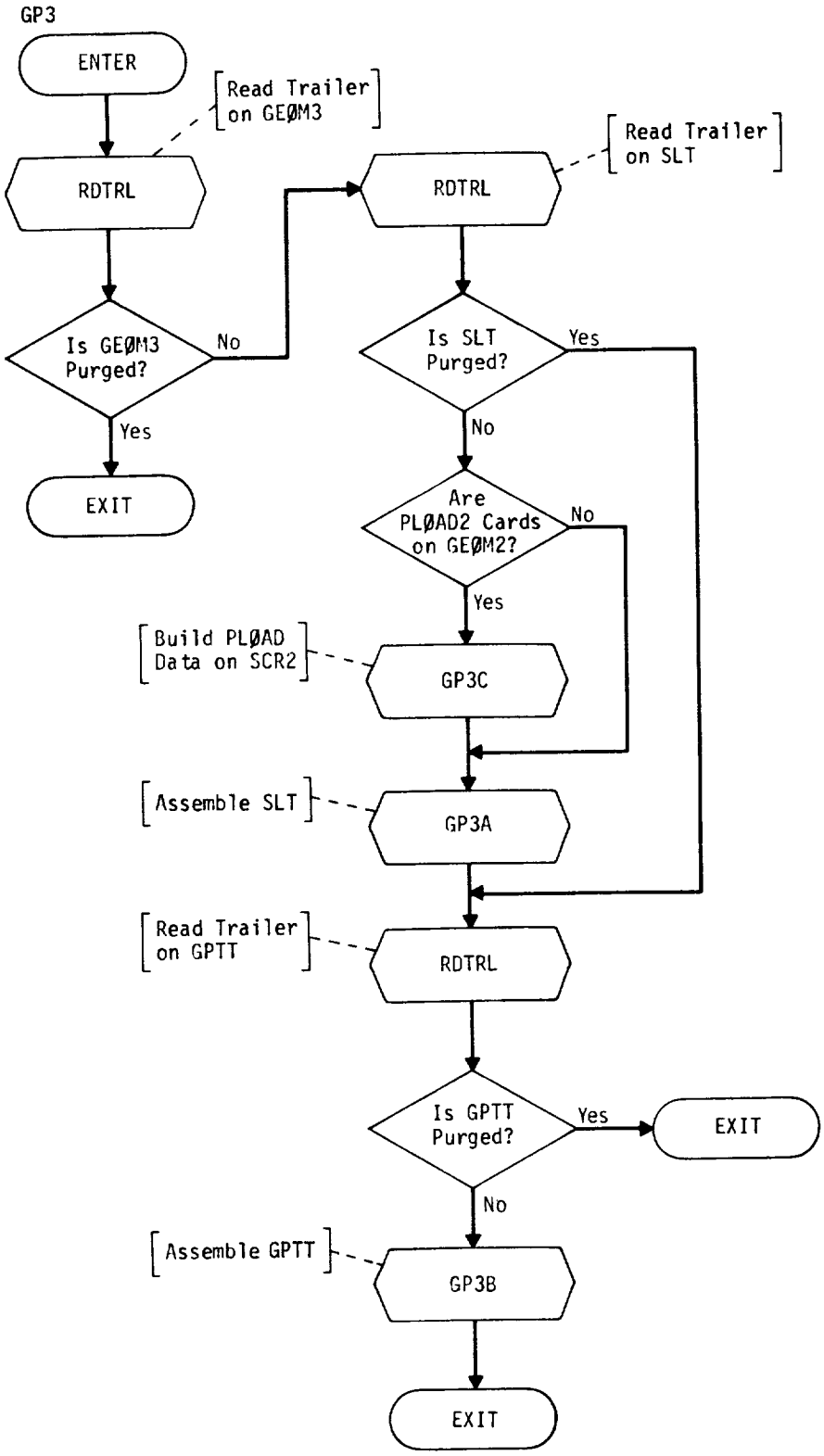


Figure 1. Flowchart for module GP3

4.26 FUNCTIONAL MODULE TA1 (TABLE ASSEMBLER)

4.26.1 Entry Point: TA1

4.26.2 Purpose

TA1 processes element connection data, element property data and geometry. These data are merged in two different sorts for efficiency in later processing. The Element Summary Table contains, for each element, connection, property and geometry data. The Element Connection and Properties Table contains, for each grid or scalar point in the model, connection, property and geometry data for all elements connected to the point. Element temperature data are also included for both data blocks where applicable. Additionally, general elements are processed and the GEI (General Element Input) data block is assembled.

4.26.3 DMAP Calling Sequence

```
TA1,    ECT,EPT,BGPDT,SIL,GPTT,CSTM/EST,GEI,ECPT,GPCT/V,N,LUSET/V,N,NSIL/V,N,NØSIMP
        /C,N,0/V,N,NØGENL/V,N,GENEL  $
```

4.26.4 Input Data Blocks

ECT - Element Connection Table.

EPT - Element Properties Table.

BGPDT - Basic Grid Point Definition Table.

SIL - Scalar Index List.

GPTT - Grid Point Temperature Table.

CSTM - Coordinate System Transformation Matrices.

Note: The ECT, BGPDT and SIL data blocks may not be purged.

4.26.5 Output Data Blocks

EST - Element Summary Table.

GEI - General Element Input.

ECPT - Element Connection and Properties Table.

GPCT - Grid Point Connection Table.

Note: No output data block may be purged.

4.26.6 Parameters

LUSET - Input-integer-no default. Degrees of freedom in the g-displacement set.

NSIL - Output-integer-no default. Number of grid points plus number of scalar points in the model.

NØSIMP - Output-integer-no default. Number of elements in the model (exclusive of general elements) or -1 if no elements.

NØGENL - Output-integer-no default. Number of general elements in the model or -1 if no general elements.

GENEL - Output-integer-no default. GENEL = -NØGENL.

4.26.7 Method

4.26.7.1 General Comments

The purpose of the Table Assembler Module is to combine all of the element data in a convenient form for the generation of the structural matrices (ECPT) and for the calculation of the element stresses and forces (EST). The complete description of an element requires: (1) the locations of the connected grid points, (2) necessary orientation data and end conditions, (3) element properties, (4) a material reference, (5) transformations from the basic system to the global coordinate system and (6) element temperature. Scalar elements require no geometric or material data. General elements may require geometric data.

Four data blocks are formed in this module. The ECPT data block is used in structural matrix generation. It contains all element data for each grid point or scalar point in the order of the sequenced grid point numbers (internal grid point indices). The EST data block contains element data in groups of element type and with sequential element I.D. numbers within each group. It is used to calculate element stresses and forces in a convenient order for output. The GEI data block contains the general element flexibility and support matrices. The GPCT data block is used to allocate storage in the structural matrix assemblers.

The reason for assembling the ECPT and EST tables rather than generating functions such as element stiffness matrices and stress functions is the expected size of the problems. The computing time used to recalculate certain data is expected to be compensated for by the time savings that result from sorting and merging smaller tables.

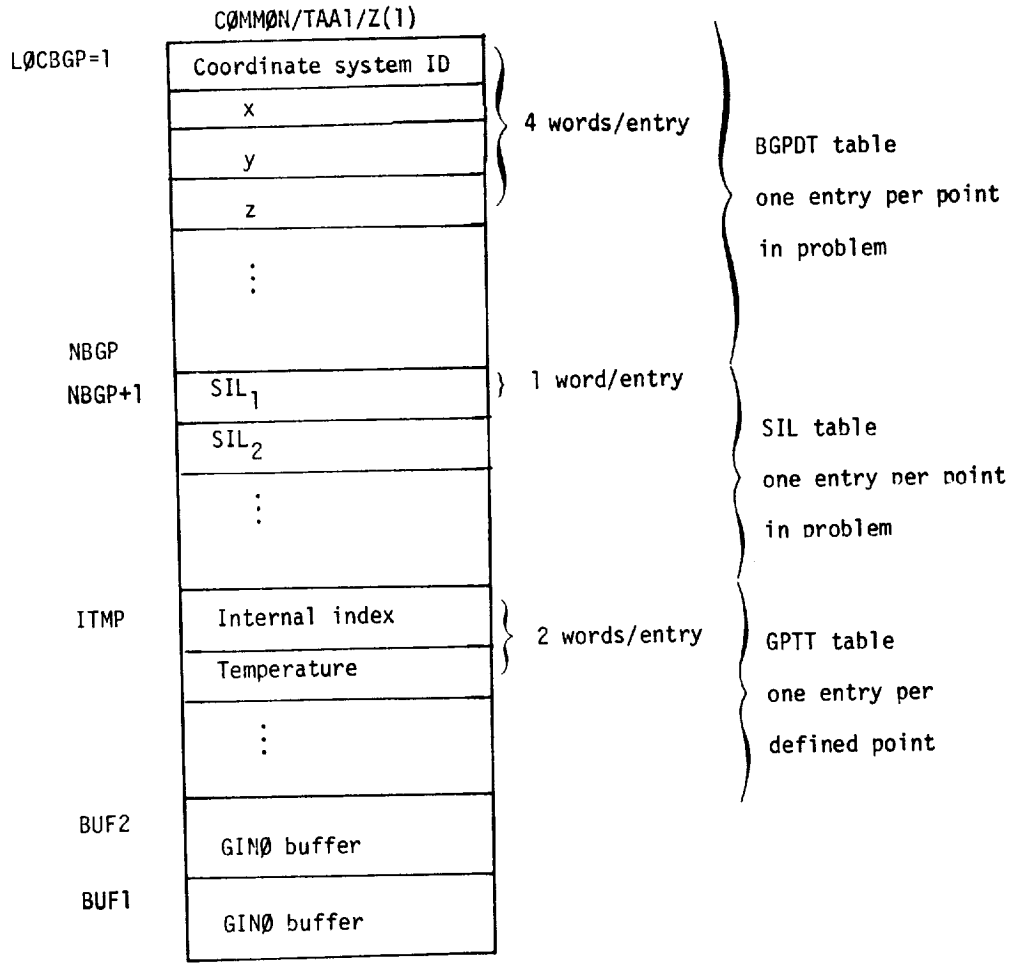
Subroutine TA1 is the main control program for the module. It executes each of the major routines of the Table Assembler (TA1A to assemble the EST, TA1B to assemble the ECPT and GPCT, and TA1C to assemble the GEI) depending on the status of the data blocks and data for the problem. A flow chart of TA1 is included as Figure 1.

4.26.7.2 TA1A

Assembly of the Element Summary Table is performed in two steps. For the first step, the EPT is read into core one property type at a time. The ECT is read one element at a time. For each element the referenced property data are found by performing a binary search in the EPT in core. The ECT and EPT data are written on SCR1, a scratch file, one element at a time. one logical record per element type.

To initiate the second step, the BGPDT and SIL data blocks are read into core. If a temperature set is selected, the appropriate temperature data from the GPTT are read into core. Data from SCR1 are read one element at a time. Internal indices for the grid points are used as pointers into the BGPDT and SIL tables. The temperature of the element is computed by averaging the temperatures of the referenced grid points. Each temperature is found by performing a binary search in the GPTT with the internal index of the grid point. If the entry is not found in the GPTT, the default temperature for the set is substituted. The internal indices are now replaced with corresponding scalar index values. A line comprising ECT, EPT, BGPDT and GPTT data for the element is written on the EST. Each logical record of the EST comprises all data of one element type.

Allocation of core storage during the second step is as follows:



4.26.7.3 TA1B

The ECPT data block is assembled in TA1B. Each logical record in the ECPT corresponds to a grid or scalar point in the model. For each point, the data for each element connected to the point are listed. The data for each element are identical to the EST data. Each set of element data will be listed in the ECPT "n" times, where "n" is the number of grid points connected by the element. A sample of the ECPT is given in Table 1. The logical phases of the operation are as follows:

A list is formed in core giving the relative locations of the elements in the ECT data in the order which they will be placed in the ECPT data block. Storage is allocated by

forming the GPC (Grid Point Counter) and then replacing the GPC by a running sum of elements connected to points. A sample is:

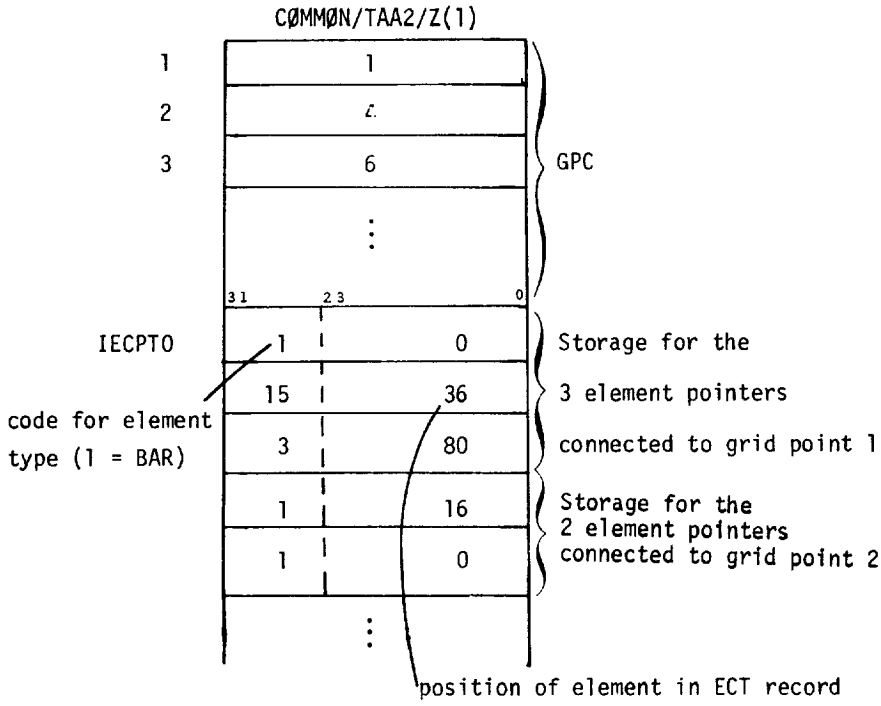
Implied Grid Point internal index	GPC Number of Connected Elements	GPCS Sum of Previous Elements
(1)	5	0
(2)	3	5
(3)	1	8
(4)	2	9
(5)	6	11
.	.	17
.	.	.
.	.	.
.	.	.

The GPC is formed as follows. An area of core equal to the number of grid and scalar points of the model is set to zero. The ECT is read one element at a time. The storage location corresponding to the internal index of each referenced grid point is incremented by one. When each element in the ECT has been processed, a running sum of the core table is formed.

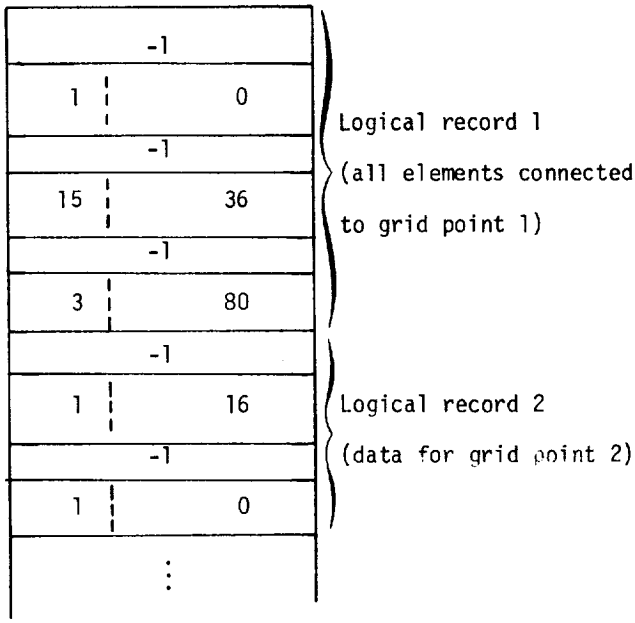
The contents of each word in the GPC now provide a pointer to the first storage location where a second pointer to the element data will be stored. Since the total number of connected elements may exceed available core storage, spill logic is provided. A band of entries in the GPC is determined. The ECT is read one element at a time. The position of each element of a given element type is determined by summing the number of words for each entry for the element (i.e. if m = number of words per ECT entry, then the position of the element in the ECT record = (i-1)*m where i = entry number in the ECT record). For each point referenced by the element (which is in the band currently being processed), the contents of the associated position in the GPC is fetched. The element position and its type are stored at the indicated locations.

The location in the GPC is incremented by one. When a pass of the ECT is complete, the skeleton ECPT is written, one logical record per point in the band of the current pass. Each

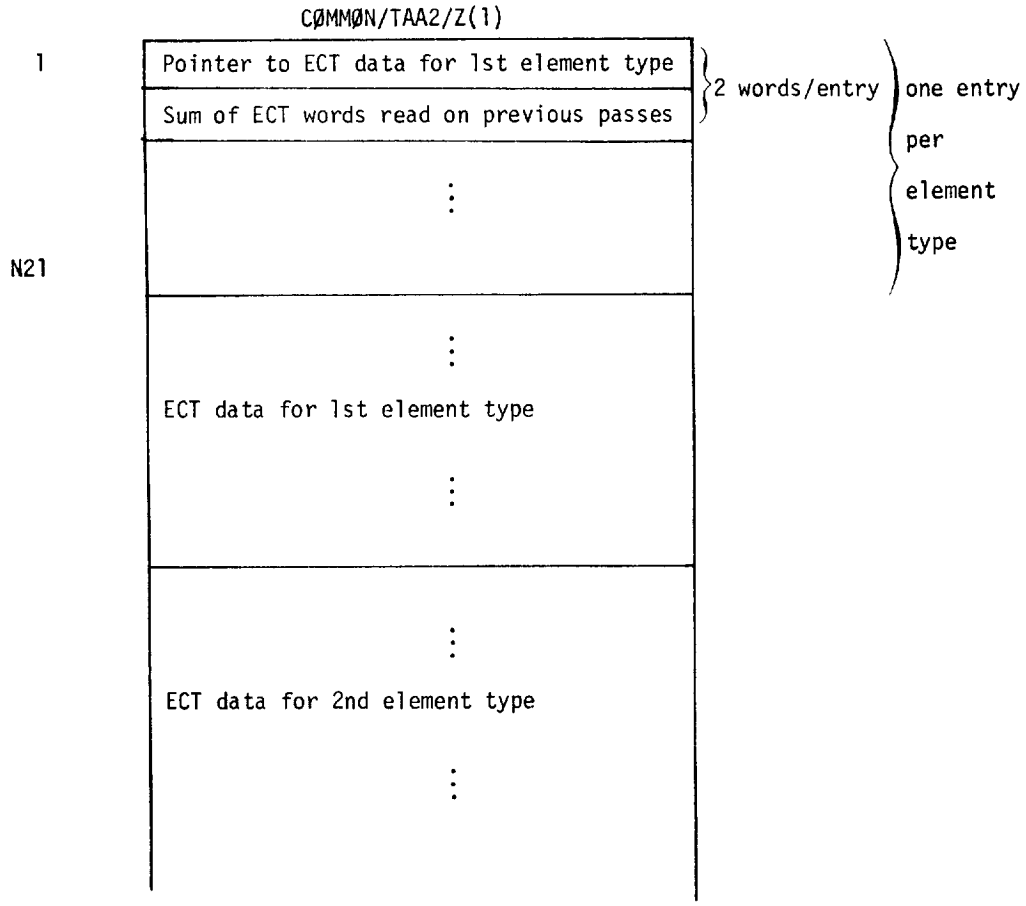
logical record consists of pairs of (-1, element pointer). The number of pairs equals the number of elements connected to the point. Example:



Skeleton ECPT data is written on a scratch file:



When all entries of the GPC have been processed, the skeleton ECPT is complete. An area of core equal to two words per element type is set to zero. The ECT is read into core following the above table until either the ECT is exhausted or core is filled. A pointer to the beginning of data for each element type is stored in the table. The skeleton ECPT is now read one pair of words at a time. If the data to which the element pointer points are currently in core, they are written out on a second scratch file. The first word of the entry contains the number of words in the ECT data. If the data are not in core, the pair (-1, element pointer) is written on the second scratch file. This process is repeated until the skeleton has been exhausted. If all the ECT is in core, the new skeleton ECPT is complete. Otherwise, the files for old and new skeleton ECPT are switched, and the process continues by reading more of the ECT data into core. Storage allocation at this point follows:



The remainder of the ECPT assembly is very similar to the EST construction. The principal difference is that the entire EPT is held in core. A pointer table similar to the ECT table is formed (two entries per element type). The EPT is read into core, and the first position of each EPT type is stored in the table along with the number of EPT entries of that type. If the BGPDT, SIL and GPTT can be held in core with the EPT, the ECPT is assembled in one pass. Otherwise, two passes are made. The skeleton ECPT is read one entry at a time. For each element, the property data are attached by performing a binary search in the associated property table in core. If one pass, the BGPDT, SIL and GPTT data are attached as in TA1A. Otherise the ECT and EPT data are written on a scratch file, and a second pass is made to attach the BGPDT, SIL and GPTT data. Table 1 contains sample ECPT contents.

During the final pass of the ECPT assembly, the GPCT is constructed. The GPCT is comprised of one logical record per point (same as ECPT). Each logical record consists of the pivot point and all other points connected to the pivot by means of element connections.

Table 1. Sample ECPT Data Contents.

Reference Grid Point First Scalar Index (Pivot Point)	235
1st Element Type	9 (Triangular membrane)
Connected Grid Point First Scalar Indices (From ECT and SIL)	625 235 535
Anisotropic Angle (From ECT)	0.0
Material Number (From EPT)	2
1st Element Properties (From EPT)	0.5 0.0 1.0
Location and Orientation Data for Grid Points (From BGPDT)	5, 10.0, 100.0, 0.0 5, 11.0, 100.0, 0.0 0, 235.0, 50.0, 25.0
Element Temperature (From GPTT)	15.5
2nd Element Type	34 (Bar)
. . etc. . .	
Last Element Type (for point 235)	1 (Rod)
. . etc. . .	
End of Logical Record (new grid point)	******
Reference Grid Point First Scalar Index (Pivot Point)	241
. . etc. .	

4.26.7.3 TA1C

The General Element flexibility and support matrices are assembled in TA1C. The data, given in the ECT data block, consist of the following sections for each General Element:

1. A list of the independent degrees of freedom, u_i, in terms of grid points and components and/or scalar points.
2. A list of supporting degrees of freedom, u_d, given by grid and scalar points (May be null).
3. A flexibility matrix [Z] with rows and columns corresponding to the list of given u_i points.
4. A support matrix [S] with rows corresponding to the u_i points and columns corresponding to the u_d points. (May be null).

The tasks of TA1C are to (1) convert the lists of u_i and u_d to scalar indices and sort them by increasing scalar index, (2) rearrange the matrices to correspond to the sorted lists of u_i and u_d degrees of freedom, and (3) on user option calculate the support matrix from grid point geometry.

These tasks are accomplished as follows:

1. For each set of coordinates (u_i and u_d) of length n, a 4xn table is formed where the four entries corresponding to each degree of freedom are:
   a) The position as given (1, 2, 3 ...n)
   b) The internal position (zero initially)
   c) The grid or scalar point I.D. (Internal index)
   d) The grid point component (1 = x, 2 = y, etc)
2. The list is sorted on the third and fourth position. If a point I.D. in the third position is duplicated, the duplicates are sorted on the components in the fourth position. The list now corresponds to the desired order of increasing scalar indices.
3. The SIL data block is read, and each of the points in the list is converted to its SIL value. (The position of a SIL number is its internal point index.) The SIL value and the component c_i determine the scalar index, N_i, of a degree of freedom by:

$$N_i = SIL + (c_i - 1) \quad (1)$$

for grid points and

$$N_i = SIL \quad (2)$$

for scalar points.

The list of scalar indices is written on the GEI data block file.

4. In order to rearrange the matrices to correspond to the proper sequence of degrees of freedom the above list is modified as follows:

   a) The internal position number is placed in the second position of each entry in the list. The first entry uses 1, the second, 2, etc.

   b) The list is sorted again to return the original order as given on the input card images. The first position of each entry supplies this order. The internal position of a term in a matrix is now given by the second numbers in each entry.

5. Steps (1) through (4) are repeated for both the u_i and u_d sets.

6. The [Z] and [S] matrices are rearranged according to the sorted lists of degrees of freedom. The row and column numbers are converted by the algorithm:

   a) For a term of Z_{ij}, where i and j are the row and column as given by the matrix order, the position i of the u_i irternal number list gives ℓ, the new column number.

   b) For a term of S_{ij}, where i and j are the external row and column numbers, the row number i is converted using the u_i list. The column number j uses the u_d list.

   If a matrix is small enough to fit in core, the new row and column numbers are used to place the term in its correct position in core. If the matrix will be larger than core, the new row and column indices and the term itself are written on a scratch file. When all terms are processed, the file is sorted to form a sequenced matrix. The terms of the matrix are written on the GEI file in full matrix form.

4.26.7.4 TA1CA

The [S] matrix must be generated if the user inputs a list of six u_d points and does not supply an [S] matrix. This is accomplished in subroutine TA1CA as follows:

The BGPDT and CSTM data are read into core. (SIL is already in core).

A six by six matrix $[D_o]$ is formed, where each row corresponds to a u_d scalar index (j). $[D_o]$ is a six by six matrix which transforms the three translations and three rotations in the basic coordinate system to the six rigid body u_d degrees of freedom:

$$\{u_d\} = [D_o] \begin{Bmatrix} x_o \\ y_o \\ z_o \\ \theta_{xo} \\ \theta_{yo} \\ \theta_{zo} \end{Bmatrix}_{\text{basic}} \tag{3}$$

The steps for generation of each row of [D] are as follows:

1. X_j, Y_j, Z_j the BGPDT location vector for the grid point containing scalar point j is found.
2. $[T_j]$ the 3x3 global-to-basic transformation matrix for the grid point containing j is fetched using subroutine MAT.
3. If scalar j is a translation, define:

$$[E_j] = \begin{bmatrix} 1 & 0 & 0 & 0 & Z_j & -Y_j \\ 0 & 1 & 0 & -Z_j & 0 & X_j \\ 0 & 0 & 1 & Y_j & -X_j & 0 \end{bmatrix}. \tag{4}$$

The column of $[T_j]$ corresponding to the degree of freedom j is defined as the row vector $\{V_j\}^T$. The row vector of $[D_o]$ corresponding to point j is:

$$\{D_{oj}\}^T = \{V_j\}^T [E_j]. \tag{5}$$

4. If scalar j is a rotation, the column of $[T_j]$ corresponding to the degree of freedom j is defined as the row vector $\{V_j\}^T$. The row vector of $[D_o]$ corresponding to point j is:

$$\{D_{oj}\}^T = \{0\ 0\ 0:\ V_j^T\}. \tag{6}$$

When all 6 rows $[D_o]$ have been generated, the matrix is inverted:

$$[H_{do}] = [D_o]^{-1}. \tag{7}$$

$[H_{do}]$ transforms the u_d displacements to rigid body motions about the basic coordinate system, i.e.:

$$\left\{\begin{array}{c} X_o \\ Y_o \\ Z_o \\ \theta_{xo} \\ \theta_{yo} \\ \theta_{zo} \end{array}\right\}_{\text{basic}} = [H_{do}]\{u_d\} \tag{8}$$

If the matrix is ill-conditioned, a fatal error exists.

The [S] matrix may now be calculated a row at a time. The list of u_i points, (s_i), is read one at a time, and the [S] matrix is formed a row at a time. Call each row $\{S_i\}^T$.

The steps for generation of each row are as follows:

1. Using the basic coordinates X_i, Y_i, Z_i for the grid point corresponding to scalar s_i, the global-to-basic transformation matrix $[T_i]$ is fetched.

2. The column of $[T_i]$ corresponding to the scalar coordinate of u_i is defined as the row vector $\{V_i\}^T$.

3. If u_i is a translation, we form

$$[E] = \begin{bmatrix} 1 & 0 & 0 & 0 & Z_i & -Y_i \\ 0 & 1 & 0 & -Z_i & 0 & X_i \\ 0 & 0 & 1 & Y_i & -X_i & 0 \end{bmatrix} , \quad (9)$$

and

$$\{S_i\}^T = \{V_i\}^T [E] [H_{od}] . \quad (10)$$

4. If u_i is a rotation:

$$\{S_i\}^T = \{0\ 0\ 0 : V_i^T\} [H_{od}] . \quad (11)$$

4.26.8 Subroutines

4.26.8.1 Subroutine Name: TA1

1. Entry Point: TA1
2. Purpose: Module driver.
3. Calling Sequence: CALL TA1

4.26.8.2 Subroutine Name: TA1A

1. Entry Point: TA1A
2. Purpose: To assemble the EST.
3. Calling Sequence: CALL TA1A

4.26.8.3 Subroutine Name: TA1B

1. Entry Point: TA1B
2. Purpose: To assemble the ECPT and GPCT.
3. Calling Sequence: CALL TA1B

4.26.8.4 Subroutine Name: TA1C

1. Entry Point: TA1C

2. Purpose: To assemble the GEI.

3. Calling Sequence: CALL TA1C

4.26.8.5 Subroutine Name: TA1CA

1. Entry Point: TA1CA

2. Purpose: To calculate the general element support matrix [S] .

3. Calling Sequence: CALL TA1CA

4.26.8.6 Subroutine Name: TA1F

1. Entry Point: TA1F

2. Purpose: To assemble the super quadrilateral portion of the SEST (Force Method only).

3. Calling Sequence: CALL TA1F

4.26.8.7 Subroutine Name: TA1G

1. Entry Point: TA1G

2. Purpose: To assemble the final EST (Force Method only).

3. Calling Sequence: CALL TA1G

4.26.9 Design Requirements

4.26.9.1 Allocation of Core Storage

TA1A. Step (1): Maximum core storage equals all property data for one element type plus three GINØ buffers.

Step (2): Maximum core storage equals 5* (number of grid and scalar points in model) plus 2* (number of grid point temperatures in selected set) plus two GINØ buffers.

TA1B. The initial steps of TA1B are open ended. The final assembly of the ECPT requires that all EPT data be held in core at one time. At another time the same storage requirement as TA1A exists plus additional storage to hold a list of the maximum number of points connected to any one point by means of element connections.

TA1C. The maximum storage requirement equals 5* (number of grid and scalar points in the model) plus the CSTM table plus 4* (number of u_i + number of u_d points) plus three GINØ buffers.

4.26.9.2 Environment

TA1 is designed to allow each of the major phases of the module to be in a separate overlay segment. Open core for each is defined as follows:

TA1A: /TAA1/

TA1B: /TAA2/

TA1C: /TAC1/

GINØ file names and DMAP parameters are communicated through blank CØMMØN. No block data program is used. Communication between TA1C and TA1CA occurs through /TA1CAX/ and /TAC1/.

4.26.10 Diagnostic Messages

The following messages may be issued by TA1:

2010, 2011, 2013, 2014, 2015, 2044, 2045, 2063, 2082

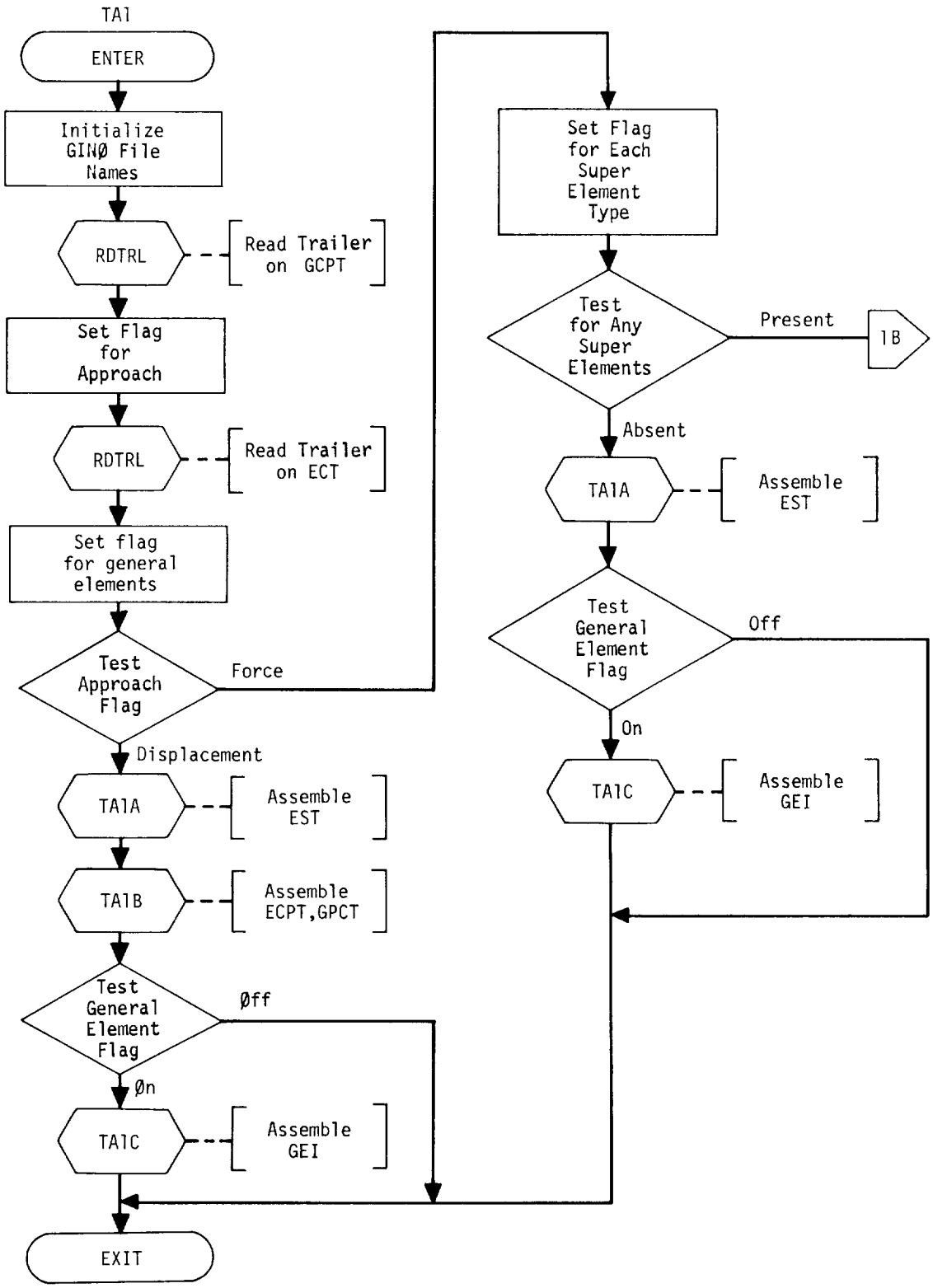


Figure 1(a). Flowchart for module TA1

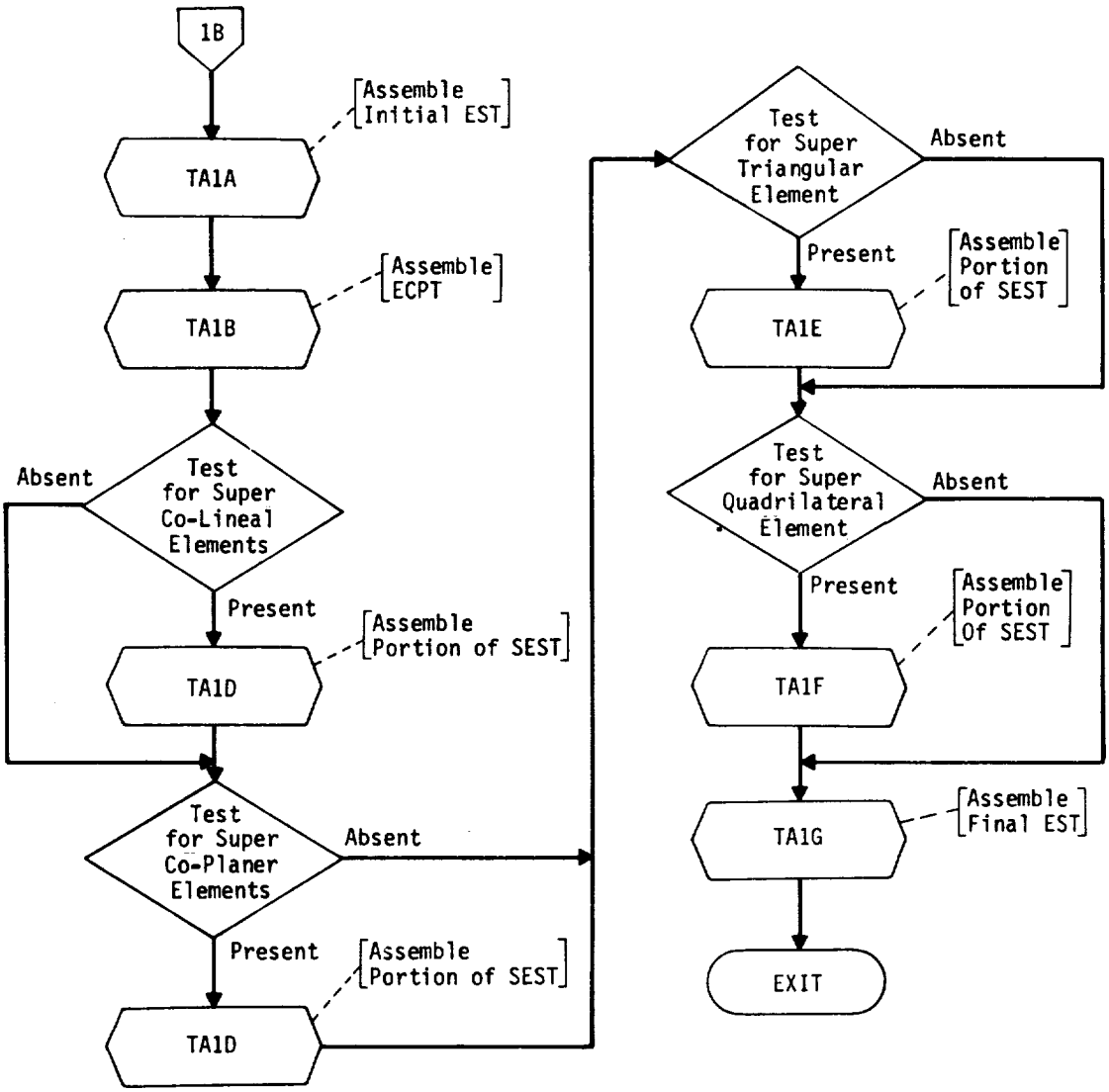


Figure 1(b). Flowchart for module TA1

4.27 FUNCTIONAL MODULE SMA1 (STRUCTURAL MATRIX ASSEMBLER - PHASE 1)

4.27.1 Entry Point: SMA1

4.27.2 Purpose

To generate the stiffness matrix exclusive of general elements, $[K^{x}_{gg}]$, the structural damping matrix, $[K^{4}_{gg}]$, and the Grid Point Singularity Table, GPST.

4.27.3 DMAP Calling Sequence

```
SMA1    CSTM,MPT,ECPT,GPCT,DIT/KGGX,K4GG,GPST/V,N,NØGENEL/V,N,NØK4GG    $
```

4.27.4 Input Data Blocks

CSTM - Coordinate System Transformation Matrices.

MPT - Material Properties Table.

ECPT - Element Connection and Properties Table.

GPCT - Grid Point Connection Table.

DIT - Direct Input Tables.

Notes: 1. The CSTM may be purged.

2. The ECPT and the GPCT cannot be purged, or else a fatal error will occur.

3. If some element references a material property, the MPT cannot be purged.

4. If some material property is temperature dependent, DIT cannot be purged.

4.27.5 Output Data Blocks

KGGX - Partition of stiffness matrix exclusive of general elements - g set.

K4GG - Partition of structural damping matrix - g set.

GPST - Grid Point Singularity Table.

Notes: 1. Neither KGGX or GPST may be pre-purged.

2. If K4GG is pre-purged, K4GG will not be generated.

3. If NØGENL > 0 (see below) the GPST will not be generated.

4.27.6 Parameters

NØGENL - Input-integer-no default value. NØGENL is the number of general

elements in the model. If NØGENL > 0 then GPST will not be generated.

NØK4GG - Output-integer-no default value. If K4GG has been pre-purged or is the zero matrix, NØK4GG is set equal to -1. Otherwise NØK4GG is set = +1.

4.27.7 Method

Matrix generation modules such as SMA1, SMA2, DSMG1 and PLA4 all use the ECPT (or a variation thereof in the case of PLA4) and its companion data block, the GPCT, as the basic data blocks for generation of stiffness and structural damping matrices (SMA1), mass and viscous damping matrices (SMA2), the differential stiffness matrix (DSMG1), and the non-linear stiffness matrix (PLA4). The central role of the ECPT data block in these modules is discussed in section 1.8.

Subroutine SMA1 is the module driver. Its tasks are: to set up GINØ buffers and matrix control blocks for the output matrices; to determine if the CSTM data block exists, and, if it does, to read it into open core and call the initialization routine PRETRD; to call the material properties initialization routine PREMAT, where material property cards and tables are read into open core; to open and position all files so that input data blocks are ready to be read and output data blocks are ready to be written; to call subroutine SMA1A, the module "workhorse", which will create the output data blocks. Upon return from SMA1A, files are closed and trailers are written. Subroutine descriptions for PRETRD and PREMAT can be found in section 3.4.37 and 3.4.36 respectively.

Subroutine SMA1A consists entirely of a loop in which, during each pass of the loop, a record of the GPCT and a record of the ECPT are processed in a complementary manner. Each pass through this principal loop creates either one or six rows (or columns since $[K^{x}_{gg}]$ and $[K^{4}_{gg}]$ are symmetric of the stiffness matrix, KGGX, (and the structural damping matrix, K4GG, if called for in the DMAP calling sequence). One row will be generated if the pivot point, the first word of both the GPCT record and the ECPT record, is a scalar point; six rows will be generated if the pivot point is a grid point. The latter case holds in the majority of cases. The loop is terminated when an end-of-file is sensed on the file containing the GPCT.

The loop begins by attempting to read the first two words of the current GPCT record. If the second non-standard return from subroutine READ occurs, it implies the current pivot point has no elements connected to it so that one or six null rows must be output for the

matrices. This non-standard return, it should be noted, does not occur in the majority of cases. A normal return from READ implies a normal path through the principal loop. The remainder of the GPCT record is read into core, and a pointer table is constructed. This pointer table relates the scalar index numbers of the GPCT record, which are the column indices of the (possible) non-zero terms of the generated matrices, to locations in core in which the corresponding submatrices will reside. The pointer table is defined recursively as follows:

$$p_1 = 1 \tag{1}$$

$$p_i = p_{i-1} + q, \; i > 1 \tag{2}$$

where q is 6 if the point (i-1) is a grid point and q is 1 if the point (i-1) is a scalar point.

It is then determined if all the submatrices corresponding to the current pivot point can be held in open core. If they can, there is no problem. If they cannot, spill logic is provided only if the structural damping matrix is not called for. Rather than compute all submatrices, store them on a scratch file, retrieve them when computations have been completed and sort them, the following approach was adopted. It is determined what the maximum number of rows that can be held in core is: 3, 2 or 1. The element submatrices are computed in their respective routines (i.e. KRØD, KBAR) and passed to the "insertion" subroutine, SMA1B, which "inserts" into the correct open core positions only those rows which can be contained.

When the ECPT record is exhausted, the link vector, LINK, (see discussion below of the LINK variable in /SMA1CL/) is searched to determine if any structural element entries in the ECPT were skipped because the corresponding element subroutine did not reside in the link (element subroutine overlay segment) currently in core. If there did exist such an element, the ECPT file is backspaced and the ECPT record is reprocessed such that all element submatrices not computed on the previous pass(es) of the ECPT record will be computed. If (or when) the LINK array is identically zero, signifying that all element submatrices corresponding to the current pass of the ECPT record have been computed, the DETCK subroutine is called if the input parameter NØGENL ≤ 0. DETCK generates the GPST by examining the "translational" and "rotational" diagonal 3 X 3 submatrices of the 6 rows of the KGGX matrix currently in core. If the total number of rows to be computed (6 or 1) for the current ECPT record is not in core due to spill problems, DETCK stores those elements of the 3 X 3 sub-

matrices currently in core in local variables and then processes the entire 3x3 submatrices on the last pass of the ECPT record (see definitions of the variables LRØWIC and NRØWSC in /SMA1CL/ below).

After DETCK returns, the number of rows in core are packed onto the KGGX (and, if called for, the K4GG) data block(s) using the standard matrix packing routines BLDPK, ZBLPKI and BLDPKN. If the last row in core is not equal to the total number of rows to be computed, the ECPT file is backspaced and the record is processed again, this time the next set of 3, 2 or 1 rows being output. If the last row in core is equal to the total number of rows to be computed, the processing of the ECPT record is complete and a transfer is made to the top of the "GPCT and ECPT processing" loop to process the next record of the GPCT and ECPT. The loop terminates when an end-of-file is encountered while attempting to read the GPCT. Upon loop termination, SMA1A returns to SMA1.

It should be noted that the most difficult logic of the routine involves the LINK vector and the spill logic. The programmer is advised that the LINK vector logic will not be used on any of the current hardware/software configurations because 1) the routine residing in segment (link) 2, KCØNE, cannot be used in conjunction with any other structural element routine and 2) the axisymmetrical element routines KTRIRG, KTRAPR and KTØRDR cannot (from a mathematical modeling point of view) be used in conjunction with any other structural element routines. The spill logic is very seldom entered since for the majority of cases the geometry of the mathematical model is such that the number of words in any GPCT record - and hence the number of (potentially) non-zero columns in any six rows of the matrix - is generally quite small. A high upper limit for the number of words in any GPCT record would be 40.

4.27.7.1 Determining Grid Point Singularities in Subroutine DETCK

Let the pivot point be a grid point with scalar index p in the following discussion. Let [Q] be the "translational" or "rotational" 3x3 symmetric submatrix along the diagonal of the stiffness matrix, $[K^{x}_{gg}]$, i.e., the rows and columns of the "translational" [Q] matrix would correspond to scalar index numbers p, p+1, and p+2; and the rows and columns of the "rotational" [Q] matrix would correspond to scalar index numbers p+3, p+4, and p+5.

The following steps comprise the algorithm for determining the presence or absence of grid point singularities. The discussion assumes [Q] is the "translational" 3x3 matrix but the same algorithm holds for the "rotational" [Q].

1. The matrix [Q] is scaled by the magnitude of the largest term, Q_{max}:

$$[B] = \frac{[Q]}{Q_{max}}. \tag{3}$$

If the largest term is non-positive, the singularity is of order 3, and the scalar index numbers p, p+1 and p+2 are written on the GPST.

2. The vector magnitudes of 3x1 columns (rows) are calculated:

$$b_1 = \sqrt{B_{11}^2 + B_{12}^2 + B_{13}^2}, \tag{4}$$

$$b_2 = \sqrt{B_{21}^2 + B_{22}^2 + B_{23}^2}, \tag{5}$$

$$b_3 = \sqrt{B_{31}^2 + B_{32}^2 + B_{33}^2}. \tag{6}$$

3. For each $b_i = 0$, the singularity order counter IØRDER is increased by one.

4. If two b_i are zero, the order of the singularity is two, and the scalar index numbers j and k corresponding to these two rows of [B] are written on the GPST.

5. If one b_i is zero, and i is the row such that $b_i = 0$, define j and k as the other rows of [B] and calculate:

$$m = \det \begin{bmatrix} B_{jj} & B_{jk} \\ B_{kj} & B_{kk} \end{bmatrix} , \tag{7}$$

$$R = \left(B_{jj}^2 + B_{kj}^2\right)\left(B_{jk}^2 + B_{kk}^2\right) . \tag{8}$$

If $\frac{m}{R} < 10^{-2}$, the order of the singularity is 2 and the GPST contains the paired scalar index values for i, j, k in the order (1) (i,j) if $B_{kk} > 0$ or (2) (i,k) if $B_{jj} > 0$. If $\frac{m}{R} \geq 10^{-2}$, the order of the singularity is one and only the SIL value for i is written on the GPST.

6. If all $b_i > 0$, we calculate

$$D = \det [B] \tag{9}$$

If $D > .5 \times 10^{-2} \times (b_1 b_2 b_3)$, there are no singularities, and DETCK returns if [Q] is the "rotational" matrix, or, if [Q] is the "translational" matrix, the "rotational" [Q] is input to the algorithm.

7. If $D < .5 \times 10^{-2} \times (b_1 b_2 b_3)$, one or more singularities exist. The following terms are calculated:

$$m_1 = \det \begin{bmatrix} B_{22} & B_{23} \\ B_{32} & B_{33} \end{bmatrix} , \tag{10}$$

$$m_2 = \det \begin{bmatrix} B_{11} & B_{13} \\ B_{31} & B_{33} \end{bmatrix} , \tag{11}$$

$$m_3 = \det \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} , \tag{12}$$

$$R_1 = \sqrt{(B_{22}^2 + B_{23}^2)(B_{31}^2 + B_{33}^2)}\ , \tag{13}$$

$$R_2 = \sqrt{(B_{11}^2 + B_{13}^2)(B_{31}^2 + B_{33}^2)}\ , \tag{14}$$

$$R_2 = \sqrt{(B_{11}^2 + B_{12}^2)(B_{21}^2 + B_{22}^2)}\ . \tag{15}$$

8. Determine i, j, k such that:

$$\frac{m_i}{R_i} > \frac{m_j}{R_k} > \frac{m_k}{R_k}\ . \tag{16}$$

9. If $\frac{m_i}{R_i} < 10^{-2}$, the singularity is of order 2. Redefine i, j, k such that $B_{ii} < B_{jj} < B_{kk}$. The SIL values for the paired indexes (j,k), (i,k) and (i,j) are written on GPST only if the corresponding B is greater than zero. For instance if B_{kk} is zero, the SIL pair (i,j) is not written on the GPST.

10. If $\frac{m_i}{R_i} < 10^{-2}$, (see step 8) the singularity is of order 1. The SIL values are written on the GPST in the order

$$i \text{ since } \frac{m_i}{R_i} > 10^{-2}\ , \tag{17}$$

$$j \text{ if } \frac{m_i}{R_j} > 10^{-2}\ , \tag{18}$$

$$k \text{ if } \frac{m_i}{R_k} > 10^{-2}\ . \tag{19}$$

4.27.8 Subroutines

The utility routines PRETRD and PREMAT are called in SMA1 for initialization purposes so that the structural element subroutines can call the entry points TRANSD of PRETRD and MAT of PREMAT to fetch Coordinate System Transformation Matrices (CSTM) data and material properties data respectively. GMMATD is used by element routines as a general matrix multiply routine, and INVERD is used for inversion of small in-core (order usually $\leq$ 12) matrices. It should be noted that all matrices referenced in the structural element subroutines are stored by rows and are double precision. See the subroutine descriptions for these routines in section 3.

The principal means of communicating an element entry of the ECPT to an element stiffness matrix generation routine is through /SMA1ET/. This fact is not explicitly stated in each of the descriptions of the element routines (e.g. KRØD) given below. Since much of the mathematics needed for generating: (1) element stiffness matrices (module SMA1); (2) element mass and damping matrices (module SMA2); (3) element contributions to load vectors (module SSG1); (4) element stress (and force) data recovery (module SDR2); (5) element differential stiffness matrices (module DSMG1); (6) element stress (and force) data recovery for non-linear elements in a Piecewise Linear Analysis Rigid Format problem (module PLA3); and (7) element stiffness matrices for non-linear elements in a Piecewise Linear Analysis problem (module PLA4), is similar or even identical, detailed mathematical algorithims are grouped by element in section 4.87.

It should be noted that routines DKI,DKK,DKM,DKINT,DKJ,DKEF,DK89,DK100,KFAC,DKJAB, DK219,DK211,RØMBDK,D4K,D5K,D6K and DMATRX are used only (directly or indirectly) by the axisymmetric shell element routines KTRIRG, KTRAPR and KTØRDR.

4.27.8.1 Subroutine Name: SMA1

1. Entry Point: SMA1
2. Purpose: See discussion above.
3. Calling Sequence: CALL SMA1

4.27.8.2 Subroutine Name: SMA1A

1. Entry Point: SMA1A

2. Purpose: See discussion above.

3. Calling Sequence: CALL SMA1A

4.27.8.3 Subroutine Name: SMA1B

1. Entry Point: SMA1B

2. Purpose: This routine, called by the module's element stiffness matrix generation routines such as KRØD, KBAR, etc., adds a double precision 6 x 6 or 1 x 1 matrix, $[K^e]$, to the "submatrix" of $[K^x_{gg}]$ or $[K^4_{gg}]$ corresponding to the current pivot point.

3. Calling Sequence: CALL SMA1B (KE,J,II,IFILE,DAMPC)

KE - Row-stored double precision 6 x 6 or 1 x 1 matrix to be added to the submatrix in core - double precision - input.

J - The column index of the $[K^x_{gg}]$ or $[K^4_{gg}]$ matrix which corresponds to the first column of the $[K^e]$ matrix - integer - input.

II - If II is 0, the $[K^e]$ matrix is 6 x 6. If II is greater than zero, it is the row index of the $[K^x_{gg}]$ or $[K^4_{gg}]$ matrix corresponding to the 1 x 1 matrix $[K^e]$ to be added - integer - input.

IFILE - GINØ file number of the matrix in core being added to - KGGX or K4GG - integer - input.

DAMPC - If $[K^e]$ is 6 x 6 and the $[K^4_{gg}]$ matrix is called for, the input matrix $[K^e]$ is multiplied by DAMPC before being added to the submatrix of $[K^4_{gg}]$ in core.

4.27.8.4 Block Data Program Name: SMA1BD

1. Entry Point: SMA1BD

2. Purpose: Block data program which sets GINØ file numbers, I/Ø parameters, and SMA1 overlay parameters.

3. Calling Sequence: None

4.27.8.5 Subroutine Name: DETCK

1. Entry Point: DETCK

2. Purpose: This routine generates the Grid Point Singularity Table by examining the 3x3 "translational" and "rotational" diagonal submatrices of the KGGX matrix. This routine is called after the submatrix for each pivot point has been completed.

3. Calling Sequence: CALL DETCK (JARG)

JARG - { If JARG = 0, the pivot point has elements connected to it.
If JARG = -1, the pivot point is a scalar point and no elements are connected to it.
If JARG = 1, the pivot point is a grid point and no elements are connected to it.

4.27.8.6 Subroutine Name: KRØD

1. Entry Point: KRØD

2. Purpose: To generate the element stiffness matrix for a RØD element.

3. Calling Sequence: CALL KRØD

4.27.8.7 Subroutine Name: KBAR

1. Entry Point: KBAR

2. Purpose: To generate the element stiffness matrix for a BAR element.

3. Calling Sequence: CALL KBAR

4.27.8.8 Subroutine Name: KTUBE

1. Entry Point: KTUBE

2. Purpose: To generate the element stiffness matrix for a TUBE element.

3. Calling Sequence: CALL KTUBE

4.27.8.9 Subroutine Name: KPANEL

1. Entry Point: KPANEL

2. Purpose: To generate the element stiffness matrix for a SHEAR or TWIST panel element.

3. Calling Sequence: CALL KPANEL (IARG)

IARG - { IARG = 4 calls for generation of the matrix for a shear panel;
{ IARG = 5 implies a twist panel.

4.27.8.10 Subroutine Name: KTRMEM

1. Entry Point: KTRMEM

2. Purpose: To generate the element stiffness matrix for a TRMEM element.

3. Calling Sequence: CALL KTRMEM (I)

I = { 0 - Do complete triangular membrane.
{ 1 - Return 3 transformed 3 x 3 matrices only for pivot point. If I = 1, KTRMEM is called by KQDMEM.

4.27.8.11 Subroutine Name: KQDMEM

1. Entry Point: KQDMEM

2. Purpose: To generate the element stiffness matrix for a QDMEM element.

3. Calling Sequence: CALL KQDMEM

4.27.8.12 Subroutine Name: KTRBSC

1. Entry Point: KTRBSC

2. Purpose: To generate the element stiffness matrix for a basic bending triangle element.

3. Calling Sequence: CALL KTRBSC (I)

I = { 0 - Do complete element computation for basic bending triangle
{ 1 - Form only the $[K^U]$ 9x9 matrix.
{ 2 - Form only the $[K^U]$ 9x9 matrix but save the $[H]^{-1}$ and [S] matrices.

4.27.8.13 Subroutine Name: KTRPLT

1. Entry Point: KTRPLT

2. Purpose: To generate the element stiffness matrix for a triangular plate element.

3. Calling Sequence: CALL KTRPLT

4.27.8.14 Subroutine Name: KQDPLT

1. Entry Point: KQDPLT

2. Purpose: To generate the element stiffness matrix for a quadrilateral plate element.

3. Calling Sequence: CALL KQDPLT

4.27.8.15 Subroutine Name: KTRIQD

1. Entry Point: KTRIQD

2. Purpose: To generate the element stiffness matrix for any of the following elements: TRIA1,TRIA2,QUAD1,QUAD2.

3. Calling Sequence: CALL KTRIQD (IARG)

$$IARG = \begin{cases} 1 - \text{TRIA1 element.} \\ 2 - \text{TRIA2 element.} \\ 3 - \text{QUAD1 element.} \\ 4 - \text{QUAD2 element.} \end{cases}$$

4.27.8.16 Subroutine Name: KELAS

1. Entry Point: KELAS

2. Purpose: To generate stiffness matrix contributions from the ELAS1,ELAS2,ELAS3 and ELAS4 elements and structural damping matrix contributions from the ELAS1,ELAS2 and ELAS3 elements.

3. Calling Sequence: CALL KELAS (IARG)

IARG - Indicates the type of element being processed. It can take on the values 1,2,3 and 4 denoting the ELAS1,ELAS2,ELAS3 and ELAS4 elements respectively. Integer-input.

4.27.8.17 Subroutine Name: KBEAM

1. Entry Point: KBEAM

2. Purpose: To generate the element stiffness matrix for a BEAM element.

3. Calling Sequence: CALL KBEAM

4.27.8.18 Subroutine Name: KCØNE

1. Entry Point: KCØNE

2. Purpose: To generate the element stiffness matrix for a conical shell problem.

3. Calling Sequence: CALL KCØNE

4.27.8.19 Subroutine Name: KTRIRG

1. Entry Point: KTRIRG

2. Purpose: To calculate an element stiffness matrix for a triangular cross-section ring, TRIARG, element.

3. Calling Sequence: CALL KTRIRG

4.27.8.20 Subroutine Name: KTRAPR

1. Entry Point: KTRAPR

2. Purpose: To calculate an element stiffness matrix for a trapezodial cross-section ring, TRAPRG, element.

3. Calling Sequence: CALL KTRAPR

4.27.8.21 Subroutine Name: KTØRDR

1. Entry Point: KTØRDR

2. Purpose: To calculate an element stiffness matrix for a toroidal thin shell ring, TØRDRG, element.

3. Calling Sequence: CALL KTØRDR

4.27.8.22 Function Name: DKI

1. Entry Point: DKI

2. Purpose: To evaluate integrals in double precision for the triangular and trapezoidal cross-section rings in subroutines KTRIRG and KTRAPR.

3. Calling Sequence: DP = DKI (I,J,K,L,M,N,IP,IQ,R,Z)

I, J - The subscripts of R defining two lines on the limit of integration, integer-input.

K, L - The subscripts of R, Z defining another line on the limit of integration, integer-input.

M, N - The subscripts of R, Z defining the fourth line on the limit of integration, integer-input.

IP,IQ - Integers that define the power of the r and z variables respectively, - input.

R,Z - Vectors of the r and z coordinates of all points used to describe the area of integration, double precision - input.

4.27.8.23 Function Name: DKK

1. Entry Point: DKK
2. Purpose: To calculate the slope of a line given two points in function DKI.
3. Calling Sequence: DP = DKK (I,J,R,Z)

   I, J - The subscripts of R, Z defining the two points.

   R, Z - Vectors of the r and z coordinates.

4.27.8.24 Function Name: DKM

1. Entry Point: DKM
2. Purpose: To calculate the y-intercept of a line given two points in function DKI.
3. Calling Sequence: DP = DKM (I,J,R,Z)

   I, J - The subscripts of R, Z defining the two points.

   R, Z - Vectors of the r and z coordinates.

4.27.8.25 Function Name: DKINT

1. Entry Point: DKINT
2. Purpose: To evaluate the following function in the FØRTRAN function routine DKI:

$$f_1(A,B) = \frac{1}{W} \sum_{t=0}^{W} C\emptyset EF \cdot A^t \cdot B^{W-t} \cdot AJ \tag{20}$$

where

$$C\emptyset EF = \begin{cases} \prod_{s=1}^{t} \frac{W-s+1}{s} & \text{for } t \neq 0 \\ 1 & \text{for } t = 0 \end{cases} \tag{21}$$

and

$$AJ = \begin{cases} \frac{R(J)^{(W+V-t+1)} - R(I)^{(W+V-t+1)}}{W+V-t+1} & \text{for } (W+V-t+1) \neq 0 \\ \ln\left[\frac{R(J)}{R(I)}\right] & \text{for } (W+V-t+1) = 0 . \end{cases} \tag{22}$$

3. Calling Sequence: DP = DKINT (I,J,A,B,V,W,R,Z)

   I, J - The subscripts of R, Z.

   A, B - The arguments of the function f in Equation 20.

   V, W - Integer parameters of the function.

   R, Z - Vectors of the r and z coordinates.

4.27.8.26 Function Name: DKJ

1. Entry Point: DKJ

2. Purpose: To evaluate the following function in function DKINT.

$$DKJ = \begin{cases} \frac{R(J)^{(s+1)} - R(I)^{(s+1)}}{s+1} & \text{for } s+1 \neq 0 \\ \ln\left[\frac{R(J)}{R(I)}\right] & \text{for } s+1 = 0 . \end{cases} \tag{23}$$

3. Calling Sequence: DP = DKJ (I,J,R,S)

   I, J - The subscripts of R

R - Vector of the r coordinates.

S - Integer parameters of the function.

4.27.8.27 Function Name: DKEF

1. Entry Point: DKEF

2. Purpose: To evaluate the following function in function DKINT.

$$DKEF = \begin{cases} \prod_{s=1}^{t} \frac{w-s+1}{s} & \text{for } t \neq 0 \\ 1 & \text{for } t = 0. \end{cases} \tag{24}$$

3. Calling Sequence: DP = DKEF (T,W)

T, W - Integer parameters of the function.

4.27.8.28 Function Name: DK89

1. Entry Point: DK89

2. Purpose: To evaluate the following function in function DKI.

$$DK89(I,A,B) = \frac{1}{B^{M+1}} \sum_{s=0}^{M} M! \, (-A)^s \cdot d \ , \tag{25}$$

where

$$d = \begin{cases} \frac{(A+B\cdot R(I)^{(M+1-N-S)}}{(M-S)!\ S!(M+1-N-S)} & \text{for } (M+1-N-S) \neq 0 \\ \frac{\ln(|A + B\cdot R(I)|)}{(M+1-N)!\ (N-1)!} & \text{for } (M+1-N-S) = 0 \ . \end{cases} \tag{26}$$

3. Calling Sequence: DP = DK89 (I,A,B,M,N,R)

I - The subscript of R.

A, B - The arguments of the function.

M, N - Integer parameters of the function.

R - Vector of the r coordinates

4.27.8.29 Function Name: DK100

1. Entry Point: DK100

2. Purpose: To evaluate the following function in subroutine DKI.

$$DK100(I,A,B) = \frac{-1}{A^{(M+N-1)}} \sum_{s=0}^{M+N-2} (M+N-2)! \cdot d \quad , \tag{27}$$

where

$$d = \begin{cases} \dfrac{(A+B\cdot R(I))^{(M-1-S)} \cdot (-B)^{S}}{(M+N-2-S)!\ S!\ (M-1-S)\cdot R(I)^{(M-1-S)}} & \text{for } (M-1-S) \neq 0 \\ \dfrac{(-B)^{M-1} \cdot \ln\left(\left|\dfrac{A+B\ R(I)}{R(I)}\right|\right)}{(M-1)!(N-1)!} & \text{for } (M-1-S) = 0 \ . \end{cases} \tag{28}$$

3. Calling Sequence: DP= DK100 (I,A,B,M,N,R)

I - The subscript of R.

A, B - The arguments of the function.

M, N - Integer parameters of the function.

R - Vector of the r coordinates.

4.27.8.30 Function Name: KFAC

1. Entry Point: KFAC

2. Purpose: To evaluate the factorial function in functions DK89 and DK100.

3. Calling Sequence: K = KFAC (N)

N - The integer argument of the function.

If N < 2, the functional value is set to 1.

4.27.8.31 Function Name: DKJAB

1. Entry Point: DKJAB

2. Purpose: To evaluate the following function in FØRTRAN function DKI using the function evaluated in FØRTRAN function DK89.

$$DKJAB(I,A,B) = \frac{R(I)^{M} \ln(|A+B\cdot R(I)|)}{M} - \frac{B}{M} \cdot DK89(I,A,B) \ . \tag{29}$$

3. Calling Sequence: DP= DKJAB (I,A,B,M,N,R)

I - The subscript of R.

A, B - The arguments of the function.

M, N - Integer parameters of the function.

R - Vector of the r coordinates.

4.27.8.32 Function Name: DK219

1. Entry Point: DK219

2. Purpose: To evaluate the following function in FØRTRAN function DKI using the function evaluated in FØRTRAN function DK100.

$$DK219\ (I,A,B) = -\frac{\ln(|A+B\cdot R(I)|)}{M\cdot R(I)^M} + \frac{B}{M}\cdot DK100\ (I,A,B) \tag{30}$$

3. Calling Sequence: DP= DK219 (I,A,B,M,N,R)

I - The subscript of R.

A, B - The arguments of the function.

M, N - Integer parameters of the function.

R - Vector of the r coordinates.

4.27.8.33 Function Name: DK211

1. Entry Point: DK211

2. Purpose: To evaluate the following function in FØRTRAN function DKI.

$$DK211(I,A,B) = \begin{cases} 0 & \text{for } B\cdot R(I) = A \\ \frac{1}{2}[\ln(|2\cdot B\cdot R(I)|)]^2 & \text{for } B\cdot R(I)\neq A,\ [B\cdot R(I)]^2=A^2 \\ [\ln|A|]\cdot[|n|R(I)|] - \sum_{t=1}^{\infty} \frac{1}{t^2}\left[\frac{-B\cdot R(I)}{A}\right]^t & \text{for } [B\cdot R(I)]^2 < A^2 \\ \frac{1}{2}[\ln(|B\cdot R(I)|)]^2 + \sum_{t=1}^{\infty} \frac{1}{t^2}\left[\frac{-A}{B\cdot R(I)}\right]^t & \text{for } [B\cdot R(I)]^2 > A^2 \end{cases} \tag{31}$$

3. Calling Sequence: DP = DK211 (I,A,B,R)

I - The subscript of R.

A, B - The arguments of the function.

R - Vector of the r coordinates.

4.27.8.34 Subroutine Name: RØMBDK

1. Entry Point: RØMBDK

2. Purpose: To evaluate integrals in double precision for the toroidal thin shell ring in subroutine KTØRDR.

3. Calling Sequence: CALL RØMBDK (A,B,NØSIG,PRECIS,NUM,ITDØNE,FINTG,KØDE,FUNCT,X)

A, B - Lower and upper limit of integration respectively.

NØSIG - Number of correct significant digits desired.

PRECIS - Actual number of significant digits attained.

NUM - Maximum number of halvings of the interval [A,B] to be made.

ITDØNE - Actual number of halvings of the interval [A,B].

FINTG - Resultant value of integral.

KØDE - Print control (not used).

FUNCT - Function subprogram used to evaluate the integral.

X - Vector of parameters used by function subprogram.

4.27.8.35 Function Name: D4K

1. Entry Point: D4K

2. Purpose: To evaluate the following function to be integrated by subroutine RØMBDK.

$$D4K = \frac{\phi^{J} \cdot \sin^{2}\phi}{R_1 - R_p \cdot \sin\alpha_1 + R_p \cdot \sin\alpha_1 \cdot \cos\phi + R_p \cdot \cos\alpha_1 \cdot \sin\phi} \quad . \tag{32}$$

3. Calling Sequence: DP = D4K (X)

X - Vector of function parameters.

X(1) - ϕ

X(2) - R_p

X(3) - R_1

X(4) - $\cos \alpha_1$

X(5) - $\sin \alpha_1$

X(6) - J + 1

4.27.8.36 Function Name: D5K

1. Entry Point: D5K

2. Purpose: To evaluate the following function to be integrated by subroutine RØMBDK.

$$D5K = \frac{\phi^J \cdot 2 \cdot \sin\phi \cdot \cos\phi}{R_1 - R_p \cdot \sin\alpha_1 + R_p \cdot \sin\alpha_1 \cos\phi + R_p \cdot \cos\alpha_1 \cdot \sin\phi} . \quad (33)$$

3. Calling Sequence: DP= D5K (X)

X - Vector of function parameters.

X(1) - ϕ

X(2) - R_p

X(3) - R_1

X(4) - $\cos \alpha_1$

X(5) - $\sin \alpha_1$

X(6) - J + 1

4.27.8.37 Function Name: D6K

1. Entry Point: D6K

2. Purpose: To evaluate the following function to be integrated by subroutine RØMBDK.

$$D6K = \frac{\phi^J \cos^2\phi}{R_1 - R_p \cdot \sin\alpha_1 + R_p \cdot \sin\alpha_1 \cdot \cos\phi + R_p \cdot \cos\alpha_1 \cdot \sin\phi} . \quad (34)$$

3. Calling Sequence: DP = D6K (X)

X - Vector of function parameters.

X(1) - ϕ

X(2) - R_p

X(3) - R_1

X(4) - $\cos \alpha_1$

X(5) - $\sin \alpha_1$

X(6) - J + 1

4.27.8.38 Subroutine Name: DMATRX

1. Entry Point: DMATRX

2. Purpose: To form the element stiffness matrix in field coordinates for the toroidal thin shell ring in subroutine KTØRDR.

3. Calling Sequence: CALL DMATRX (D,V,C,CA,CA2,VA,DM,DB,YI)

D	-	Resultant stiffness matrix.
V,C,CA, CA2,VA, DM,DB	-	Terms used in the evaluation of the stiffness matrix.
YI	-	Array of integral values.

4.27.9 Design Requirements

4.27.9.1 Open Core Design

The open core common block for module SMA1 is defined by the following FØRTRAN statements:

1. DØUBLE PRECISIØN DZ(1)
2. INTEGER IZ(1)
3. CØMMØN /SMA1X/ Z(1)
4. EQUIVALENCE (Z(1),IZ(1),DZ(1)).

The open core layout is given in Figure 1.

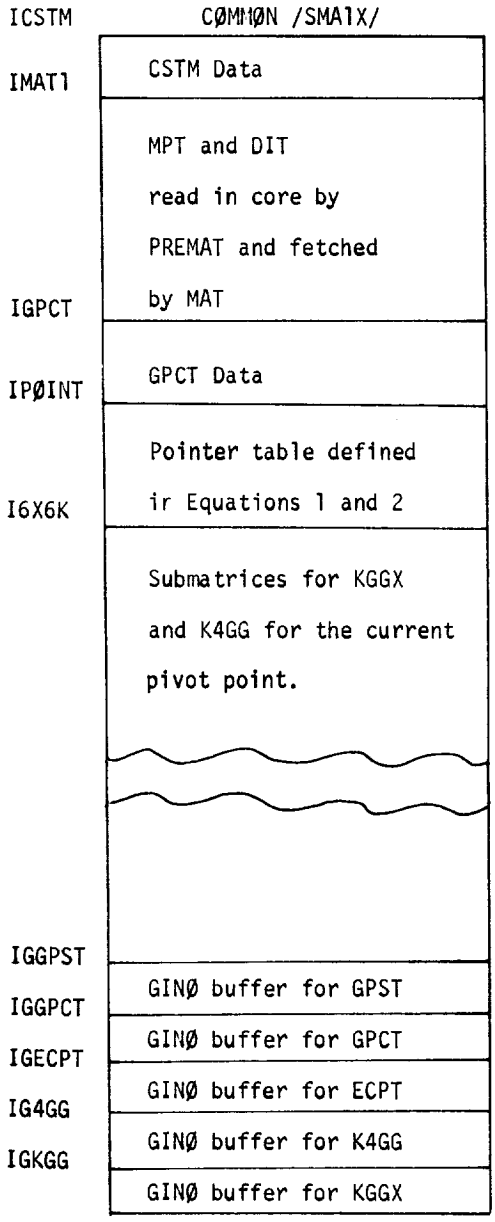


Figure 1. Open core layout for module SMA1.

The definition of the variables is as follows:

ICSTM - The zero pointer to the CSTM portion of open core; defined to be zero.

IMAT1 - The zero pointer to the MPT and DIT data read into core by subroutine PREMAT; defined to be ICSTM + NCSTM, where NCSTM is the length of the CSTM portion of open core.

IGPCT - The zero pointer to the GPCT portion of open core; defined to be IMAT1 + MATCR, MATCR being the length of open core used by subroutine PREMAT.

IPØINT - The zero pointer to the pointer table in open core. The pointer table is used as a dictionary to relate the GPCT to the submatrices in core; defined to be IGPCT + NGPCT, NGPCT being the length of the GPCT.

I6X6K - The zero pointer to the submatrices of KGGX; I6X6K = (IPØINT + NPØINT -1)/ 2+2 where NPØINT = NGPCT is the length of the pointer table. The extra arithmetic to define I6X6K is necessary because the submatrices are double precision numbers. While the above indices are single precision indices, I6X6K is a double precision index.

I6X64 - The zero pointer to be submatrices of K4GG; I6X64 = I6X6K + N6X6K, where N6X6K is the number of double precision numbers in the submatrices of KGGX. I6X64 is a double precision index.

The pointers for the GINØ buffers, IGGPST, IGGPCT, IGECPT, IG4GG and IGKGG are, unlike the above, 'one' pointers. It should be noted that the lengths NCSTM and MATCR are constant throughout the module operation, while the length of the GPCT data will vary from pivot point to pivot point as the ECPT and GPCT data blocks are processed serially. (Hence it is probable that for a pivot point with a relatively small number of elements connected to it the entire submatrix may be held in core, while spill logic will be entered only when a pivot point has a great many elements connected to it.)

4.27.9.2 Block Data Subprogram

The block data program SMA1BD sets GINØ file numbers, I/Ø parameters and SMA1 overlay parameters in common blocks /SMA1IØ/ and /SMA1CL/.

4.27.9.3 Common Storage Requirements

Blank common is used only for DMAP parameters. The following common blocks are used throughout the module: /SMA1IØ/, /SMA1BK/, /SMA1CL/, /SMA1ET/ and /SMA1DP/. They are given in detail here since other matrix assembler modules such as SMA2 and DSMG1 are designed similarly. All common block variables are integer except (1) DØDET in /SMA1CL/ which is a logical variable; (2) the array in /SMA1ET/ which is a mixed (integer and real) array; and (3) the double precision array in /SMA1DP/.

1. The SMA1IØ common block is 36 words in length and is used for SMA1 input/output parameters.

Word Number	Variable	Definition
1	IFCSTM	GINØ file number for the CSTM data block.
2	IFMPT	GINØ file number for the MPT data block.
3	IFDIT	GINØ file number for the DIT data block.
4	IDUM1	Undefined.
5	IFECPT	GINØ file number for the ECPT data block.
6	IGECPT	GINØ buffer pointer for the ECPT.
7	IFGPCT	GINØ file number for the GPCT data block.
8	IGGPCT	GINØ buffer pointer for the GPCT.
9-10	IFGEI,IGGEI	Undefined.
11	IFKGG	GINØ file number for the KGGX data block.
12	IGKGG	GINØ buffer pointer for KGGX.
13	IF4GG	GINØ file number for the K4GG data block.
14	IG4GG	GINØ buffer pointer for K4GG.
15	IFGPST	GINØ file number for the GPST data block.
16	IGGPST	GINØ buffer pointer for the GPST.
17	INRW	Input with rewind option for subroutine ØPEN.
18	ØUTRW	Output with rewind option for subroutine ØPEN.
19	CLSNRW	Close without rewind option for subroutine CLØSE.
20	CLSRW	Close with rewind option for subroutine CLØSE.
21	NEØR	No end-of-record indicator for subroutine READ.
22	EØR	End-of-record indicator for subroutine READ.
23-29	MCBKGG	Matrix control block for the KGGX matrix.
30-36	MCB4GG	Matrix control block for the K4GG matrix.

2. The SMA1BK common block is 10 words in length and is used for SMA1 open core bookkeeping parameters. It contains zero pointers and lengths for the various sub-arrays in open core.

Word Number	Variable	Definition
1	ICSTM	Zero pointer to the CSTM sub-array in open core. For example the first location of this sub-array is referenced as IZ (ICSTM + 1).
2	NCSTM	Length of the CSTM sub-array in open core.
3	IGPCT	Zero pointer to the GPCT sub-array in open core.
4	NGPCT	Length of the GPCT sub-array.
5	IPØINT	Zero pointer to the PØINT sub-array in open core.
6	NPØINT	Length of the PØINT sub-array.
7	I6X6K	Zero pointer to the 6x6 submatrices of KGGX.
8	N6X6K	Number of words allocated to the 6x6 submatrices of KGGX.
9	I6X64	Zero pointer to the 6x6 submatrices of K4GG.
10	N6X64	Undefined.

3. The SMA1CL common block is 133 words in length and is used for module control parameters.

Word Number	Variable	Definition
1	IØPT4	Indicator used by element routines denoting whether or not the K4GG matrix will be generated. IØPT4 = 0, implies no generation; IØPT4 = 1 implies generation.
2	K4GGSW	Indicator set to -1 initially. If IØPT4 = 1, then element routines will set K4GGSW = 1, when a non-zero element structural damping matrix is generated.
3	NPVT	The scalar index which is the pivot point. This is the first word of every record of the ECPT data block.
4	LEFT	The number of words of open core remaining after all sub-arrays in open core have been allocated.
5	FRØWIC	The first row of the submatrices in core. If all six rows of the matrices to be generated cannot be held in core, spill logic is initiated, and 3, 2 or 1 rows of the submatrices are generated during each pass of the ECPT record for the pivot point which causes the spill. FRØWIC can take on the values 1,2,3,4,5 or 6.
6	LRØWIC	The last row of the submatrices in core. LRØWIC is defined as FRØWIC + NRØWSC -1, where NRØWSC is the number of rows in core. If there are no spill problems, then LRØWIC = 6 if the pivot point is a grid point, and LRØWIC = 1 if the pivot point is a grid point.
7	NRØWSC	The number of rows of the submatrices currently in core.
8	TNRØWS	Total number of rows of the submatrices to be generated. TNRØWS = 6 if the pivot point is a grid point and TNRØWS = 1 if the pivot point is a scalar point. This definition holds whether or not the K4GG matrix is to be generated. (In actuality, if the K4GG is generated the total number of rows

Word Number	Variable	Definition
		generated for any ECPT record is 12 or 2).
9	JMAX	The number of columns of KGGX (and K4GG) to be generated with the current ECPT record.
10	NLINKS	The number of machine links (overlay segments) necessary to contain the module's element routines. Currently NLINKS is defined to be 3. This variable is used in conjunction with the LINK array defined below. For machines with large memories, it is desirable to have all element routines in one link, for when in any ECPT record there are elements which reside in different links, overlay overhead can be very costly (particularly on second generation computing systems).
11-50	IØVRLY	Array of length 40 which defines in which link an element routine resides. IØVRLY(I) = 1,2 or 3 defines the link, 1, 2 or 3, in which the Ith element routine resides. Currently element numbers 2 and 35, the BEAM and the CØNE, reside in link 2; element numbers 36,37 and 38, the TRIARG, TRAPRG and the TØRØDRG reside in link 3; all other elements, except element numbers 31,32,33,39 and 40 which are not defined for the displacement method, reside in link 1.
51-90	LINK	Before the current ECPT record is read for the first time, the LINK array is set to -1 for LINK(I), I = 1, ...NLINKS. When the first element is read from the ECPT, the proper element routine is called, thereby loading the link in which that element routine resides. The variable LINCØR, the link in core, is defined as LINCØR = IØVRLY(ITYPE), where ITYPE is the element's internal number, e.g., RØD = 1, BEAM = 2, etc. For the next element read from the ECPT, it is determined in what link it resides. If it resides in the link which is in core, the element routine is called. If the routine does not reside in the link currently in core, it is determined whether (a) the link has already been processed or (b) the link has not been processed in which case a "to-be-processed-later" flag is set. For case (a) LINK (ITEMP) is 1; for case (b) LINK(ITEMP) is set = 0, where ITEMP = IØVRLY(ITYPE). When an end-of-record is sensed for the ECPT, LINK(LINCØR) is set to 1 and LINK array is searched for zeros. If there are no zeros, the processing of the ECPT record is complete. If there are zeros, that is, links to be processed, the file corresponding to the ECPT is backspaced and processing of the record is repeated.
91-130	NWØRDS	NWØRDS(ITYPE) is the number of words in an ECPT entry for an element routine whose internal number is ITYPE. For ITYPE = 1 (a RØD), NWØRDS(1) = 17, for ITYPE = 34 (a BAR), NWØRDS(34) = 42, etc.
131	NØGØ	Flag used to indicate if a user fatal error message occurred in the processing of any element. NØGØ = 1 indicates an error. Execution is terminated upon completion of the processing of the GPCT. NØGØ = 0 indicates no error. Continue execution.
132	IDETCK	Used as a first pass indicator in the DETCK subroutine. There will be multiple passes through the DETCK routine, for each ECPT record, only if there are spill problems, i.e., the total number of rows to be generated for the ECPT record will not fit in core.

133	DØDET	Logical variable which if true implies the DETCK routine will be called and if false will not be called. If the input parameter, NØGENL, is greater than zero, implying general elements exist, then DØDET is set false. Otherwise DØDET is true.

4. The common block SMA1ET is 100 words in length and is used as the means of communicating the element data from the ECPT data block to the element subroutines.

5. The common block SMA1DP defines an array of 300 double precision words. This block is used as "scratch" storage by element routines. Those variables which in most FØRTRAN programs would be local subroutine variables are defined in /SMA1DP/ by the module's element routines in order to preserve core storage and hence increase open core.

4.27.9.4 Arithmetic Considerations

All floating point arithmetic operations are carried out in double precision. Both $[K^{x}_{gg}]$ and $[K^{4}_{gg}]$ are double precision matrices.

4.27.10 Diagnostic Messages

The module has a variety of "fail-safe" error checks. If any of these checks fails, it implies an obsure program design error or a computer operating system/hardware failure. Diagnostic messages 2022, 2023, 2034 are of this type.

User fatal error messages 2025, 2026, 2031, 2032, 2033, 2035, 2036, 2037, 2038, 2039 and 2040 occur when one of the structural element routines encounters some user data which makes generation of an element matrix impossible. Examples would include a user defining a RØD or BAR element of zero length; a user defining the four points of a SHEAR panel element not in the proper cyclical order; a user defining TRPLT data so that a matrix in an algorithm is non-singular; etc. Whenever this type of user fatal error occurs, the module queues all error messages and finishes processing all the data before terminating execution with a call to PEXIT.

Detailed descriptions of these error messages can be found in section 6 of the User's Manual.

4.28 FUNCTIONAL MODULE SMA2 (STRUCTURAL MATRIX ASSEMBLER - PHASE 2)

4.28.1 Entry Point: SMA2

4.28.2 Purpose

To generate the mass matrix $[M_{gg}]$ and the damping matrix $[B_{gg}]$.

4.28.3 DMAP Calling Sequence

```
SMA2    CSTM,MPT,ECPT,GPCT,DIT/MGG,BGG/V,Y,WTMASS=1.0/V,N,NØMGG/V,N,NØBGG/V,Y,
        CØUPBAR=-1   $
```

4.28.4 Input Data Blocks

CSTM - Coordinate System Transformation Matrices.

MPT - Material Properties Table.

ECPT - Element Connection and Properties Table.

GPCT - Grid Point Connection Table.

DIT - Direct Input Tables.

Notes:

1. The CSTM may be purged.
2. If some element references a material property, the MPT cannot be purged.
3. Neither the ECPT nor the GPCT may be purged.
4. If some material property is temperature dependent, DIT cannot be purged.

4.28.5 Output Data Blocks

MGG - Partition of mass matrix - g set.

BGG - Partition of damping matrix - g set.

Notes:

1. MGG cannot be pre-purged.
2. BGG can be pre-purged.

4.28.6 Parameters

WTMASS - Input-real-default value (in DMAP calling sequence) = 1.0. WTMASS is the scalar value by which the generated mass matrix will be multiplied before the columns are packed onto the output file. WTMASS is the ratio of mass to weight.

NØMGG - Output-integer-no default value. NØMGG is set equal to -1 if the generated mass matrix is the zero matrix. Otherwise it is set = +1.

NØBGG - Output-integer-no default value. If the BGG matrix is either pre-purged or is generated as the zero matrix, NØBGG is set = -1. Otherwise it is set = +1.

CØUPBAR- Input-integer-default value = -1. If CØUPBAR = -1, "consistent" mass matrices for all BAR elements will not be generated; if CØUPBAR $\geq$ 0, "consistent" mass matrices for all BAR elements will be generated.

4.28.7 Method

SMA2 is structured similarly to module SMA1. A separate module was written to generate the mass and damping matrices in order to maximize the amount of open core space available for element matrices during matrix generation. This core space was especially critical on the development computer, the IBM 7094-7040 DCS. Since SMA2 is so similar to SMA1, the details of the similarities will not be repeated here; the differences will be pointed out. The reader is referred to the Module Functional Description (MFD) for SMA1 (section 4.27).

When all rows (or, in the case of spill, the number of rows in core) for each pivot point have been computed, each matrix element of $[M_{gg}]$ is multiplied by WTMASS before being packed onto the output data block MGG.

4.28.8 Subroutines

SMA2, like SMA1, uses the utility routines PRETRD, PREMAT and GMMATD.

The principal means of communicating an element entry of the ECPT to an element mass or damping matrix generation routine is through /SMA2ET/. This fact is not explicitly stated in each of the descriptions of the element routines (e.g., MRØD) given below.

The following list gives a correspondence between SMA1 and SMA2 routines that are used only (directly or indirectly) by the axisymmetric shell element routines TRIARG and TRAPRG. All of the SMA2 routines are the same as their SMA1 counterparts except for name. The reason for duplicating these routines with different names was to maximize open core for element matrices in SMA1 and SMA2, which both reside in the same NASTRAN link. For details on each of the routines, see the corresponding SMA1 counterpart (section 4.27.8).

SMA1	SMA2
DKI	DMI
DKK	DMK
DKM	DMM
DKINT	DMINT
DKJ	DMJ
DKEF	DMEF
DK89	DM89
DK100	DM100
KFAC	MFAC
DKJAB	DMJAB
DK219	DM219
DK211	DM211

4.28.8.1 Subroutine Name: SMA2

1. Entry Point: SMA2

2. Purpose: The module driver which parallels SMA1. For further details see the method section of the MFD for SMA1 (section 4.27.7).

3. Calling Sequence: CALL SMA2

4.28.8.2 Subroutine Name: SMA2A

1. Entry Point: SMA2A

2. Purpose: To generate $[M_{gg}]$ and $[B_{gg}]$. This routine parallels SMA1A. See the MFD for SMA1 for details on SMA1A (section 4.27.8).

3. Calling Sequence: CALL SMA2A.

4.28.8.3 Subroutine Name: SMA2B

1. Entry Point: SMA2B

2. Purpose: To add a double precision 6 by 6 or 1 by 1 matrix $[K^e]$ to the "submatrix" of $[M_{gg}]$ or $[B_{gg}]$ corresponding to the current pivot point.

3. Calling Sequence: CALL SMA2B (KE,J,II,IFILE,DUMDP).

KE,J,II are as defined for subroutine SMA1B (see section 4.27.8).

IFILE - GINØ file number of the matrix in core being added to $[M_{gg}]$ or $[B_{gg}]$-integer-input.

DUMDP - A dummy double precision argument added so that the calling sequence to SMA2B would conform to that of SMA1B.

4.28.8.4 Block Data Program Name: SMA2BD.

1. Purpose: To set GINØ file numbers, I/Ø parameters and SMA2 overlay parameters in /SMA2IØ/ and /SMA2CL/.

4.28.8.5 Subroutine Name: MRØD

1. Entry Point: MRØD

2. Purpose: To generate the element mass matrix for a RØD element.

3. Calling Sequence: CALL MRØD.

4.28.8.6 Subroutine Name: MTUBE

1. Entry Point: MTUBE

2. Purpose: To generate the element mass matrix for a TUBE element.

3. Calling Sequence: CALL MTUBE

4.28.8.7 Subroutine Name: MASSTQ

1. Entry Point: MASSTQ

2. Purpose: To generate an element mass matrix for any of the two-dimensional structural elements listed under the Calling Sequence.

3. Calling Sequence: CALL MASSTQ(IARG)

$$\text{IARG} = \begin{cases} 4 = \text{TRMEM} \\ 1 = \text{QDMEM} \\ 3 = \text{TRBSC} \\ 3 = \text{TRPLT} \\ 7 = \text{QDPLT} \\ 5 = \text{TRIA1} \\ 4 = \text{TRIA2} \\ 2 = \text{QUAD1} \\ 1 = \text{QUAD2} \\ 6 = \text{SHEAR} \\ 6 = \text{TWIST} \end{cases}$$

4.28.8.8 Subroutine Name: MBAR

1. Entry Point: MBAR

2. Purpose: To generate the "diagonal" (uncoupled) element mass matrix for a BAR element.

3. Calling Sequence: CALL MBAR

4.28.8.9 Subroutine Name: MCBAR

1. Entry Point: MCBAR

2. Purpose: To generate the "consistent" (coupled) element mass matrix for a BAR element.

3. Calling Sequence: CALL MCBAR

4.28.8.10 Subroutine Name: MCØNMX

1. Entry Point: MCØNMX

2. Purpose: To generate an element mass matrix for either of the two concentrated-mass-elements listed under Calling Sequence.

3. Calling Sequence: CALL MCØNMX(IARG)

$$\text{IARG} - \begin{cases} 1 = \text{CØNM1} \\ 2 = \text{CØNM2} \end{cases}$$

4.28.8.11 Subroutine Name: MCØNE

1. Entry Point: MCØNE

2. Purpose: To generate an element mass matrix for the axisymmetric conical shell element (CØNE).

3. Calling Sequence: CALL MCØNE

4.28.8.12 Subroutine Name: MASSD

1. Entry Point: MASSD

2. Purpose: To generate 1 by 1 element mass matrices for scalar elements MASSi, i = 1,2,3,4; and 1 by 1 element damping matrices for scalar damping elements DAMPi, i = 1,2,3,4.

3. Calling Sequence: CALL MASSD(I)

I -
- 1 - Generate element mass matrix for a MASS1 element,
- 2 - Generate element mass matrix for a MASS2 element,
- 3 - Generate element mass matrix for a MASS3 element,
- 4 - Generate element mass matrix for a MASS4 element,
- 5 - Generate element damping matrix for a DAMP1 element,
- 6 - Generate element damping matrix for a DAMP2 element,
- 7 - Generate element damping matrix for a DAMP3 element,
- 8 - Generate element damping matrix for a DAMP4 element.

4.28.8.13 Subroutine Name: MTRIRG

1. Entry Point: MTRIRG

2. Purpose: To generate an element mass matrix for a triangular cross-section ring, TRIARG, element.

3. Calling Sequence: MTRIRG.

4.28.8.14 Subroutine Name: MTRAPR

1. Entry Point: MTRAPR

2. Purpose: To generate an element mass matrix for a trapezoidal cross-section ring, TRAPRG, element.

3. Calling Sequence: CALL MTRAPR.

4.28.8.15 Subroutine Name: MTØRDR

1. Entry Point: MTØRDR

2. Purpose: To generate an element mass matrix for a toroidal thin shell ring, TØRDRG, element.

3. Calling Sequence: CALL MTØRDR.

4.28.8.16 Subroutine Name: BVISC

1. Entry Point: BVISC

2. Purpose: To generate an element damping matrix for a VISC element.

3. Calling Sequence: Call BVISC

4.28.8.17 Subroutine Name: MBEAM

1. Entry Point: MBEAM

2. Purpose: To generate an element mass matrix for a BEAM element.

3. Calling Sequence: CALL MBEAM

4.28.8.18 Subroutine Name: MCRØD

1. Entry Point: MCRØD

2. Purpose: To generate the "consistent" (coupled) element mass matrix for any of the elements listed under calling sequence.

3. Calling Sequence: CALL MCRØD (IARG)

$$IARG = \begin{cases} 1 - RØD \\ 1 - CØNRØD \\ 3 - TUBE \end{cases}$$

4.28.8.19 Subroutine Name: MTRBSC

1. Entry Point: MTRBSC

2. Purpose: To generate the "consistent" (coupled) element mass matrix for a basic bending triangle element.

3. Calling Sequence: CALL MTRBSC

4.28.8.20 Subroutine Name: MTRPLT

1. Entry Point: MTRPLT

2. Purpose: To generate the "consistent" (coupled) element mass matrix for a triangular plate element.

3. Calling Sequence: CALL MTRPLT

4.28.8.21 Subroutine Name: MQDPLT

1. Entry Point: MQDPLT

2. Purpcse: To generate the "consistent" (coupled) element mass matrix for a quadrilateral plate element.

3. Calling Sequence: CALL MQDPLT

4.28.8.22 Subroutine Name: MTRIQD

1. Entry Point: MTRIQD

2. Purpose: To generate the "consistent" (coupled) element for any of the following elements: TRIA1, TRIA2, QUAD1, QUAD2.

3. Calling Sequence: CALL MTRIQD (IARG)

IARG = 1 - TRIA1 element.
2 - TRIA2 element.
3 - QUAD1 element.
4 - QUAD2 element.

4.28.9 Design Requirements

4.28.9.1 Open Core Design

The open core design for SMA2 is the same as that in SMA1 with the exception that /SMA2X/ defines the beginning of open core and only four buffers are needed, one each for MGG, BGG, ECPT and GPCT.

4.28.9.2 Common Storage Requirements

The common storage requirements for SMA2 are similar to those in SMA1. The common blocks /SMA2IØ/, /SMA2BK/, /SMA2CL/, /SMA2ET/ and /SMA2DP/ of SMA2 correspond to /SMA1IØ/, /SMA1BK/, /SMA1CL/, /SMA1ET/ and /SMA1DP/ of SMA1. See the MFD for SMA1 (see section 4.27.9). The following differences are worthy of note.

1. In /SMA2IØ/, words 15 and 16 are undefined and words 23 through 36 define matrix control blocks for MGG and BGG.

2. /SMA2CL/ is only 131 words in length, the last two words of /SMA1CL/ being reserved for variables unique to SMA1.

4.28.9.3 Arithmetic Considerations

Floating point arithmetic operations are carried out in double precision. Both $[M_{gg}]$ and $[B_{gg}]$ are real symmetric double precision matrices.

4.28.10 Diagnostic Messages

See the diagnostic message section in the MFD for SMA1 (section 4.27.10).

4.29 FUNCTIONAL MODULE GPWG (GRID POINT WEIGHT GENERATOR)

4.29.1 Entry Point: GPWG

4.29.2 Purpose

To compute the center of mass of the structure and find the principal inertias about a given point.

4.29.3 DMAP Calling Sequence

```
GPWG     BGPDT,CSTM,EQEXIN,MGG/ØGPWG/V,Y,GRDPNT/V,Y,WTMASS  $
```

4.29.4 Input Data Blocks

BGPDT - Basic Grid Point Definition Table.

CSTM - Coordinate System Transformation Matrices.

EQEXIN - Equivalance between external grid or scalar numbers and internal numbers.

MGG - Partition of mass matrix - g set.

Notes: 1. BGPDT,EQEXIN and MGG cannot be purged.

2. CSTM must be present if some grid point of the model is not in basic coordinates.

4.29.5 Output Data Blocks

ØGPWG - Grid Point Weight Generator Output Table.

Notes: This data block cannot be purged.

4.29.6 Parameters

GRDPNT - Input-integer-default = -1. GRDPNT selects the grid point about which the inertias will be calculated. If GRDPNT is not the external ID of a geometric grid point, the basic origin is used.

WTMASS - Input-real-default = 1.0. WTMASS gives the ratio of mass to weight for the structure. All output quantities are in weight units.

4.29.7 Method

The Grid Point Weight Generator module calculates the masses, centers of gravity, and inertias of the general mathematical model of the structure. The data are extracted from the $[M_{gg}]$ matrix by using a rigid body transformation calculation. Global coordinate displacements are converted to translations and rotations about a reference point.

Because of the scalar mass effects, the total mass may have directional properties, and the center of gravity may not be a unique location. This effect is shown in the output by giving for each of the three masses its own direction and center of gravity. The inertia terms are calculated by using the directional mass effects. The axes about which the inertia terms are calculated may not intersect. However, these axes are those which provide uncoupled rotation and translation effects. This is the significance of the term "center of gravity". If the structural model has been constructed using only real masses, the three masses printed out will be equal, the center of gravity will be unique, and the axes of the inertia terms will intersect at the center of gravity.

The actual computation proceeds in 4 parts.

1. Computation of the $[D]^T$ matrix takes place in subroutine GPWG1A. Six vectors are formed which describe the six motions about the reference point. The matrix [D] formed from the vectors will describe rigid body displacements in global coordinates in terms of motion at the reference point:

$$\{\ddot{u}_g\} = [D]\{\ddot{u}_o\} \quad \text{(reference point)}. \tag{1}$$

The method of generation is as follows:

EQEXIN is placed in core and searched for GRDPNT to obtain its internal sequence number. If GRDPNT is not in EQEXIN or points to a scalar point, the basic origin is used. Assuming GRDPNT is a physical grid point, BGPDT is read to obtain $\{R_o\}$ for the reference point. The BGPDT file is then rewound. CSTM (if present) is placed in core and readied for u by subroutine PRETRS. Each grid point in the BGPDT is considered in order. If it is a scalar point, zero is stored in each of the six columns of $[D]^T$.

If it is a grid point,

$$\{r_i\} = \{R_i\} - \{R_o\} = \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \end{Bmatrix}, \tag{2}$$

is computed where $\{R_i\}$ is the location of the i^{th} grid point and $\{R_o\}$ is the location of the reference point.

The transformation matrix to the grid point,

$$[T_r] = \begin{bmatrix} 0 & r_3 & -r_2 \\ -r_3 & 0 & r_1 \\ r_2 & -r_1 & 0 \end{bmatrix} , \tag{3}$$

is formed. Subroutine TRANSS calculates the 3x3 transformation matrix $[T_i]$ from global coordinates to basic coordinates for the grid point. The matrix

$$[d] = \left[\begin{array}{c:cc} T_i^T & T_i^T T_r \\ \hdashline 0 & T_i^T \end{array}\right] , \tag{4}$$

is computed. The rows of [d] form the columns of $[D]^T$ and are packed out onto a scratch file.

2. If all points were scalar points, GPWG returns; otherwise subroutine TRANP1 is called to form [D] from $[D]^T$.

3. $[M_o]$ is computed by two calls to subroutine SSG2B,

$$[M_o] = [D]^T [M_{gg}] [D]. \tag{5}$$

4. Output quantities are computed as follows:

M_o is unpacked, output and partitioned as follows:

$$[M_o] \Longrightarrow \left[\begin{array}{c:c} \overline{M}^t & \overline{M}^{tr} \\ \hdashline \overline{M}^{rt} & \overline{M}^r \end{array}\right] \tag{6}$$

(The matrix is symmetric).

A check is made for inconsistent scalar masses. Let

$$\delta = |\det [\overline{M}^t]|, \tag{7}$$

and

$$\varepsilon = \sqrt{\Sigma(\overline{M}_{ij}^t)^2} \qquad i \neq j. \tag{8}$$

If $\frac{\epsilon}{\delta} > 10^{-3}$, the coordinates should be rotated. Otherwise [S] = [I]. If rotation is necessary, the eigenvectors of $[\bar{M}^t]$, $\{e_1\}$, $\{e_2\}$, and $\{e_3\}$, are determined by the Jacobi technique. Define

$$[S] = [\ \{e_1\}\ ,\ \{e_2\}\ ,\ \{e_3\}\ ]. \tag{9}$$

The [S] matrix is output. $[M^t]$, $[M^r]$ and $[M^{tr}]$ are computed as follows:

$$[M^t] = [S]^T\ [\bar{M}^t]\ [S], \tag{10}$$

$$[M^{tr}] = [S]^T\ [\bar{M}^{tr}]\ [S], \tag{11}$$

$$[M^r] = [S]^T\ [\bar{M}^r]\ [S]. \tag{12}$$

The following terms are calculated and output: the mass terms:

$$M_x = M^t_{11}, \tag{13}$$

$$M_y = M^t_{22}, \tag{14}$$

$$M_z = M^t_{33}, \tag{15}$$

and the "centers of gravity":

$$X_x = \frac{M^{tr}_{11}}{M_x}, \quad Y_x = \frac{-M^{tr}_{13}}{M_x}, \quad Z_x = \frac{M^{tr}_{12}}{M_x}, \tag{16}$$

$$X_y = \frac{M^{tr}_{23}}{M_y}, \quad Y_y = \frac{M^{tr}_{22}}{M_y}, \quad Z_y = \frac{-M^{tr}_{21}}{M_y}, \tag{17}$$

$$X_z = \frac{-M^{tr}_{32}}{M_z}, \quad Y_z = \frac{M^{tr}_{31}}{M_z}, \quad Z_z = \frac{M^{tr}_{33}}{M_z}. \tag{18}$$

The inertias are:

$$I_{11} = M^r_{11} - M_y\ Z_y^2 - M_z\ Y_z^2\ , \tag{19}$$

$$I_{21} = -M^r_{12} - M_z\ X_z\ Y_z\ , \tag{20}$$

$$I_{31} = -M^r_{13} - M_y\ X_y\ Z_y\ , \tag{21}$$

$$I_{22} = M^r_{22} - M_z\ X_z^2 - M_x\ Z_x^2\ , \tag{22}$$

$$I_{23} = -M^r_{23} - M_x Y_x Z_x , \tag{23}$$

$$I_{33} = M^r_{33} - M_x Y_x^2 - M_y X_y^2 . \tag{24}$$

These terms form the symmetric matrix [I].

For principle inertias eigenvalues and eigenvectors are found such that:

$$\begin{bmatrix} I^p_{11} & & \\ & I^p_{22} & \\ & & I^p_{33} \end{bmatrix} = [Q]^T [I] [Q]. \tag{25}$$

[Q] contains the normalized eigenvectors (the directions of the principal inertias), and the I^p_{ii} terms are the eigenvalues. The matrices [S] and [Q] are actually coordinate rotation matrices and show the directions of the principal masses and inertias.

4.29.8 Subroutines

Utility Subroutines PRETRS,TRANSS,TRANP1,SSG2B and GMMATS are used. See subroutine descriptions, section 3.

4.29.8.1 Subroutine Name: GPWG1A

1. Entry Point: GPWG1A

2. Purpose: To form the $[D]^T$ matrix.

3. Calling Sequence: CALL GPWG1A(PØINT,BGPDT,CSTM,EQEXIN,DT,NØGØ)

PØINT	-	External grid point id of reference point - integer - input.
BGPDT	-	GINØ file number of ØGPWG - integer - input.
CSTM	-	GINØ file number of CSTM - integer - input.
EQEXIN	-	GINØ file number of EQEXIN - integer - input.
DT	-	GINØ file number of file on which $[D]^T$ will be written - integer - input.
NØGØ	-	Flag for all scalar problem - integer - output. NØGØ = 0 implies all scalars.

4.29.8.2 Subroutine Name: GPWG1B

1. Entry Point: GPWG1B

2. Purpose: To form output quantities as given in paragraph 4 of section 4.29.7.

3. Calling Sequence: CALL GPWG1B (MØ,ØGPWG,WTMASS,IP)

   MØ - GINØ file number of $[M_o]$ matrix - integer - input.

   ØGPWG - GINØ file number of ØGPWG - integer - input.

   WTMASS - Mass to weight ratio parameter - real - input.

   IP - External grid point id of reference point (=0 if basic origin was used) - integer - input.

4.29.8.3 Subroutine Name: GPWG1C

1. Entry Point: GPWG1C

2. Purpose: To compute eigenvectors and values of a 3 by 3 matrix by the classical Jacobi method.

3. Calling Sequence: CALL GPWG1C (B,E,EV,IFLAG)

   B - 3 by 3 input matrix - real - input.

   E - 3 by 3 matrix of eigenvectors - real - output.

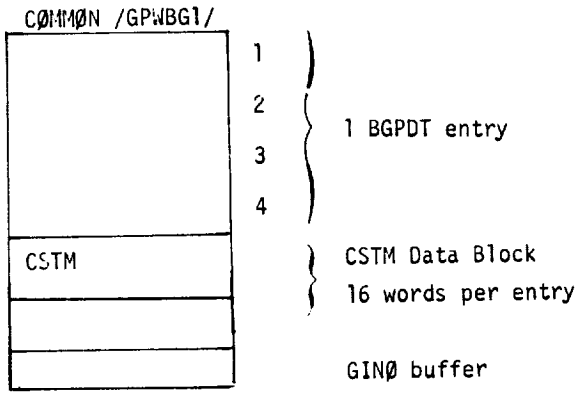
   EV - 3 eigenvalues - real - output.

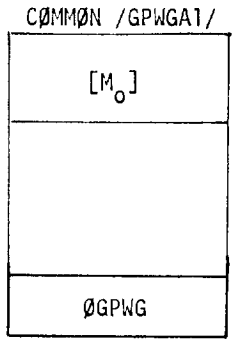
   IFLAG - Error termination flag - integer - output. If IFLAG > 0, GPWG1C could not converge.

4.29.9 Design Requirements

GPWG requires four scratch files. Open core for GPWG1A is defined at /GPWGA1/. Open core for GPWG1B is defined at /GPWGB1/.

The layout of open core is as follows:





36 words for 6 by 6 matrix

GINØ buffer

4.29.10 Diagnostic Messages

The following fatal error messages may occur: 3007, 3008.

4.30 FUNCTIONAL MODULE SMA3 (STRUCTURAL MATRIX ASSEMBLER - PHASE 3)

4.30.1 Entry Point: SMA3

4.30.2 Purpose

To generate the final stiffness matrix, $[K_{gg}]$, by generating a matrix of order g for each general element in the model, and successively adding this matrix to $[K^{x}_{gg}]$, the stiffness matrix exclusive of general elements.

4.30.3 DMAP Calling Sequence

```
SMA3    GEI,KGGX/KGG/V,N,LUSET/V,N,NØGENL/V,N,NØSIMP $
```

4.30.4 Input Data Blocks

GEI - General Element Input.

KGGX - Partition of stiffness matrix exclusive of general elements - g set.

Note:

1. GEI cannot be pre-purged.

2. KGGX may be pre-purged. This implies that the model consists entirely of general elements (i.e., there are no simple elements).

4.30.5 Output Data Blocks

KGG - Partition of stiffness matrix - g set. Contains contributions from all elements in the model, including general elements.

Note: KGG may not be pre-purged.

4.30.6 Parameters

LUSET - Input-integer-no default value. LUSET is the total number of degrees of freedom in the g displacement set. It is the order of the $[K^{x}_{gg}]$ and $[K_{gg}]$ matrices.

NØGENL - Input-integer-no default value. NØGENL is the number of general elements in the GEI data block.

NØSIMP - Input-integer-no default value. If NØSIMP < 0, $[K^{x}_{gg}]$ does not exist, i.e., all elements of the model are general elements. If NØSIMP $\geq$ 0, $[K^{x}_{gg}]$ does exist.

4.30.7 Method

4.30.7.1 Mathematical Considerations

Two matrices can be used to form a stiffness matrix for each general element in the GEI data block: a flexibility influence coefficient matrix, [Z], and a rigid body matrix, [S]. The former must be present and must be non-singular; the latter may or may not be present. The set of degrees of freedom (scalar index numbers) used by [Z] is designated the "u_i" set; the set of degrees of freedom used by [S] is designated the "u_d" set. Call the length of the u_i set m, and call the length of the u_d set n. [Z] is m by m, and [S] is m by n.

For each general element in the model, the stiffness matrix corresponding to the general element, $[K^{ge}]$, is made up of four partitions if the u_d set exists. They are:

$$[K_{ii}] = [Z]^{-1} \quad , \tag{1}$$

$$[K_{id}] = - [Z]^{-1}[S] \quad , \tag{2}$$

$$[K_{di}] = - [S]^{T}[Z]^{-1} = [K_{id}]^{T} \quad , \tag{3}$$

$$[K_{dd}] = [S]^{T}[Z]^{-1}[S] \quad . \tag{4}$$

The four matrices must be merged such that 1):

$$[K^{ge}] = \left[\begin{array}{c|c} K_{ii} & K_{id} \\ \hline K_{di} & K_{dd} \end{array}\right] \quad ; \tag{5}$$

and 2) the rows and columns of $[K^{ge}]$ must correspond to a merged list of both u_i and u_d coordinates in order of ascending scalar index numbers.

If the u_d set does not exist, then

$$[K^{ge}] = [K_{ii}] = [Z]^{-1} \quad . \tag{6}$$

4.30.7.2 Initialization

The GEI data block is opened, and the header record is skipped. It is determined whether the number of general elements, f, is even or odd. This is done to insure that the result of the final matrix addition,

$$[K_{gg}] = [K^{rs}] + [K^{gef}] , \tag{7}$$

where

$$[K^{rs}] = [K^{x}_{gg}] + \sum_{i=1}^{f-1} [K^{gei}] , \tag{8}$$

and $[K^{gef}]$ is the final general element matrix, will be written on the output data block, KGG.

If $[K^{x}_{gg}]$ does not exist, and there is only one general element, then the GINØ file number (201) for the KGG data block is stored in IFA, the variable which contains the GINØ file number of the file onto which the current $[K^{ge}]$ matrix will be packed.

The principal logic of the module driver, SMA3, is carried out in a loop in which, during each pass of the loop, a $[K^{ge}]$ matrix is generated and added, using subroutine SSG2C, to the running sum matrix, $[K^{rs}]$.

4.30.7.3 Generation and Addition of a General Element Matrix

The steps involved in the principal loop of the program are as follows:

1. The loop counter is incremented.

2. The first three words of the next logical record are read from GEI: the element id; the length of the u_i set, m; and the length of the u_d set, n.

3. The matrix control blocks for the scratch files IFB and IFC are interchanged provided that:

   a. this is not the first pass through the loop;

   b. this is not the second pass through the loop and the number of general element is odd and there are only general elements in the model;

   and

   c. this is not the second time through the loop and $[K^{x}_{gg}]$ exists.

4. It is determined whether the orders of the [Z] and [S] matrices are such that the in-core matrix routines GMMATD and INVERD can be used. This is accomplished as follows. Define

$$p = 2(m + n + m^2 + n^2 + 2mn) , \tag{9}$$

$$q = 2(m + n + m^2) + 3m , \tag{10}$$

and

$$r = \max(p,q) . \tag{11}$$

p is the number of computer words needed to store: 1) the u_i and u_d sets in two different sorts; 2) the double precision m by m [Z] matrix; 3) the double precision n by n $[S]^T[Z]^{-1}[S]$ matrix; and 4) the double precision m by n [S] matrix and the double precision m by n $[Z]^{-1}[S]$ matrix.

q is the number of computer words needed to store: 1) the u_i and u_d in two different sorts; 2) the double precision m by m [Z] matrix; and 3) 3m cells of scratch storage to be used by subroutine INVERD.

If r is less than the available amount of core, the in-core routine, SMA3A, is called to compute $[K^{ge}]$. Otherwise: 1) SMA3B generates [Z] and $-[S]^T$; 2) FACTØR decomposes [Z] into its triangular factors; 3) SSG3A computes $[Z]^{-1}$; 4) SSG2B computes $-[S]^T[Z]$; 5) TRANP1 transposes $-[S]^T$; 6) SSG2B computes $-[Z][S]^T$; 7) SSG2B computes $[S]^T[Z]^{-1}[S]$; and 8) SMA3C builds the final $[K^{ge}]$ matrix of order g by g.

5. The matrix $[K^{ge}]$ having been generated as in step 4, SSG2C is called to add $[K^{ge}]$ to $[K^{rs}]$.

4.30.8 Subroutines

Utility routines GMMATD, INVERD, FACTØR, SSG3A, SSG2B, TRANP1 and SSG2C are used in this module.

4.30.8.1 Subroutine Name: SMA3A

1. Entry Point: SMA3A

2. Purpose: To build a g by g general element matrix, $[K^{ge}]$, using the in-core matrix routines GMMATD and INVERD.

3. Calling Sequence: CALL SMA3A (MCBA)

CØMMØN /GENELY/ - see description below (section 4.30.9.2).

MCBA - The matrix control block corresponding to $[K^{ge}]$. Word 1 is input; words 2 through 7 are output.

4. Method: The u_i set and the u_d set (if present) of scalar index numbers are read into core, and a list L is formed of length m + n, such that L(k) = ℓ implies the ℓ^{th} entry of the string: $\{u_{i1},u_{i2},\ldots,u_{im},u_{d1},u_{d2},\ldots,u_{dn}\}$ is the k^{th} smallest. The m^2 single precision elements of [Z] are read and stored at double precision locations. $[Z]^{-1}$ is computed using INVERD. [S], if present, is read and stored at double precision locations. GMMATD is called twice to compute $[Z]^{-1}[S]$ and $[S]^T[Z]^{-1}[S]$. The elements of $[K^{ge}]$, as defined in Equation 5, are output to the GINØ file corresponding to MCBA(1) with non-zero terms in the row and column positions specified by the u_i and u_d sets. The list L determines the sequence of elements to be output for any one column.

4.30.8.2 Subroutine Name: SMA3B

1. Entry Point: SMA3B

2. Purpose: To create [Z] and [S] from the GEI data.

3. Calling Sequence: CALL SMA3B (IFLAG)

CØMMØN /GENELY/ - see description below (section 4.30.9.2).

IFLAG - { IFLAG = -1 implies [S] does not exist. / IFLAG = 1 implies [S] exists. } - integer - output.

4. Method: The GEI data block is read for the row numbers and non-zero terms of [Z]. These are output in standard NASTRAN matrix format by subroutine BLDPK. - $[S]^T$ is generated in a similar manner. (- $[S]^T$ is created rather than [S] for computational ease in subsequent calculations - see paragraph 4 in section 4.30.7.3 above).

4.30.8.3 Subroutine Name: SMA3C

1. Entry Point: SMA3C

2. Purpose: To create the $[K^{ge}]$ matrix from $[Z]^{-1}$, $-[Z]^{-1}[S]$, $-[S]^T[Z]^{-1}$ and $[S]^T[Z]^{-1}[S]$.

3. Calling Sequence: CALL SMA3C (IFLAG, KE)

CØMMØN /GENELY/ - see description below (section 4.30.9.2).

CØMMØN //LUSET - size of problem.

IFLAG is as described in SMA3B. Here it implies $[K^{ge}] = [Z]^{-1}$.

KE - Matrix control block for $[K^{ge}]$ - integer - input/output.

4. Method: A matrix of g size is created from $[Z]^{-1}$, $-[Z]^{-1}[S]$, $-[S]^T[Z]$ and $[S]^T[Z]^{-1}[S]$ with the non-zero terms in the row and column positions specified by the u_i and u_d lists. This matrix can be added to the existing $[K^{rs}]$ to reflect the stiffness terms of this general element.

4.30.8.4 Block Data Subprogram: SMA3BD

Purpose: To initialize the GINØ file numbers and GINØ options indicators in /GENELY/, which is discussed below.

4.30.9 Design Requirements

4.30.9.1 Open Core Design

The open core common block for the module driver SMA3 and subroutine SMA3A is defined by the following FØRTRAN statements:

1. DØUBLE PRECISIØN DQ(1)
2. INTEGER IQ(1)
3. DIMENSIØN Q(1)
4. CØMMØN /GENELX/ Q
5. EQUIVALENCE (IQ(1),DQ(1),Q(1))

SMA3 uses open core only for one GINØ buffer, which is reserved for the GEI data block while SMA3A, SMA3B, or SMA3C is executing, and which is reserved for use by SSG2C when this routine adds $[K^{ge}]$ to $[K^{rs}]$ at the end of the principal loop in the driver.

SMA3A uses low order open core as outlined in paragraph 4 in section 4.30.7 above.

The open core for subroutine SMA3B is defined at /SMAB3/ and is used for two GINØ buffers in high order open core.

The open core for subroutine SMA3C is defined at /SMAC3/ and is used for: 1) the u_i and u_d sets in low order open core and 2) six GINØ buffers in high order open core.

4.30.9.2 Common Storage Requirements

The common block /GENELY/ is used for: 1) GINØ file numbers; 2) GINØ option indicators; 3) matrix control blocks; and 4) zero pointers to sub-arrays in /GENELX/ when SMA3A executes. It is defined as follows:

```
CØMMØN /GENELY/ IFGEI,IFKGGX,IFØUT,IFA,IFB,IFC,IFD,IFE,IFF,INRW,ØUTRW,CLSRW,CLSNRW,EØR,
NEØR,MCBA(7),MCBB(7),MCBC(7),MCBE(7),MCBF(7),MCBKGG(7),IUI,IUD,IZI,IS,IZIS,ISTZIS,IBUFF3(3),
LEFT
```

Variable	Definition
IFGEI,IFKGGX,IFØUT	GINØ file numbers for the two input data blocks and the output data block respectively.
IFA	GINØ file number for the current $[K^{ge}]$ being computed.
IFB,IFC	GINØ file numbers for $[K^{rs}]$ and $[K^{rs}] + [K^{ge}]$ matrices. They are "flip-flopped" such that IFC = IFØUT for the final matrix addition.
IFD,IFE,IFF	GINØ file numbers for scratch files which are used in subroutine SMA3C.
INRW,ØUTRW,CLSRW,CLSNRW,EØR,NEØR	GINØ option indicators as defined in section 4.27.9.3.
MCBA,MCBB,...,MCBF,MCBKGG	Matrix control blocks for the matrices corresponding to IFA, IFB, ..., IFF, and IFKGXX.
IUI,IUD,IZI,IS,IZIS,ISTZIS	Zero pointers to the sub-arrays in /GENELX/ corresponding to: 1) u_i set; 2) u_d set; 3) $[Z]^{-1}$; 4) [S]; 5) $[Z]^{-1}[S]$ and 6) $[S]^T[Z]^{-1}[S]$. Note that IZI, IZIS, ISTZIS are zero pointers into double precision arrays.
IBUFF3(3)	Three word buffer which contains the general element id, m and n.
LEFT	The number of computer words currently remaining in /GENELX/.

4.30.9.3 Arithmetic Considerations

All floating point arithmetic operations are carried out in double precision.

4.30.10 Diagnostic Messages

In SMA3A, system fatal error 2028 can occur. See section 6 of the User's Manual for details.

4.31 FUNCTIONAL MODULE GP4 (GEOMETRY PROCESSOR - PHASE 4)

4.31.1 Entry Point: GP4

4.31.2 Purpose

GP4 assembles the various displacement sets and builds the displacement set definition table (USET). Additionally, for statics problems, GP4 analyzes subcases based on single-point and multipoint constraint sets, and sets parameters to control execution of the Rigid Format.

4.31.3 DMAP Calling Sequence

```
GP4     CASECC,GEØM4,EQEXIN,SIL,GPDT/RG,YS,USET/V,N,LUSET/V,N,MPCF1/V,N,MPCF2/V,N,SINGLE/
        V,N,ØMIT/V,N,REACT/V,N,NSKIP/V,N,REPEAT/V,N,NØSET/V,N,NØL/V,N,NØA  $
```

4.31.4 Input Data Blocks

CASECC - Case Control Data Table.

GEØM4 - Displacement set definitions.

EQEXIN - Equivalence between external grid or scalar and internal numbers.

SIL - Scalar Index List.

GPDT - Grid Point Definition Table.

Note: Only GEØM4 may be purged.

4.31.5 Output Data Blocks

RG - Multipoint constraint equations matrix.

YS - Constrained displacement vector(s) set.

USET - Displacement set definition table.

Note: YS may be purged.

4.31.6 Parameters

LUSET - Input-integer-no default. Degrees of freedom in the g-displacement set.

MPCF1 - Output-integer-no default. +1 if the current subcase contains multipoint constraints, -1 otherwise.

MPCF2 - Output; integer, no default. +1 if the current subcase contains a different multipoint constraint set from last the subcase, -1 if no new multipoint constraint set or no multipoint constraints in the current subcase.

SINGLE - Output, integer, no default. +1 if the current subcase contains single-point constraints, -1 otherwise.

ØMIT - Output, integer, no default. +1 if the model contains omitted coordinates, -1 otherwise.

REACT - Output, integer, no default. +1 if the model contains supports, -1 otherwise.

NSKIP - Input and output, integer, default = 0. Number of records to skip to reach the first record in the Case Control Data Block for the next subcase. (NSKIP = 0 for the first subcase).

REPEAT - Output, integer, no default. -1 if the current subcase is the last subcase in the problem, +1 otherwise.

NØSET - Output, integer, no default. -1 if MPCF1 = -1 and SINGLE = -1 and ØMIT = -1 and REACT = -1, +1 otherwise.

NØL - Output, integer, default = +1. -1 if all degrees of freedom in the model belong to dependent displacement sets (i.e. no degree of freedom belongs to an independent set), +1 otherwise.

NØA - Output, integer, default = +1. -1 if MPCF1 = -1 and SINGLE = -1 and ØMIT = -1, +1 otherwise.

4.31.7 Method

CASECC is read for each case and subcase. Parameters are set to control the return point for the next subcase. The user-requested constraint set numbers are extracted and saved for control of the following steps.

The multi-point constraint cards (MPC) each define a row of a constraint matrix equation:

$$[R_m] \{u_m\} + [R_n] \{u_n\} = 0 \qquad (1)$$

For each requested MPC set, the MPC card images are read from GEØM4, and the grid points and their scalar coordinates (or scalar points) are converted to the scalar degree of freedom

numbers. The EQEXIN and SIL data blocks are placed in core, the point "id" number is found as the first entry of a pair in the first record of EQEXIN, the corresponding number in the second entry of the pair is the internal grid point index. The i^{th} position of the SIL contains the value of the scalar degree of freedom number for the first degree of freedom of point i. The SIL value, SIL_u, for the component c, (c=1,2,...,6) of a grid point p is found by the equation:

$$SIL_u = SIL_p + (c - 1) . \qquad (2)$$

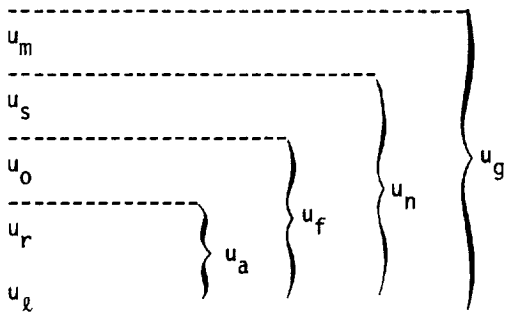
The SIL of a point is the scalar index of its first degree of freedom. The point "id's" and components given on the MPC cards are converted to scalar index numbers, a sorted list is formed of the u_m scalar indices, (see defintion below of the sets defined in USET), and the data are written on SCR1, a scratch file. Each term in the equation is paired with a packed word giving its equation number and its SIL value. The equation data are now sorted to group the data by each scalar index number in order of increasing dependent u_m point number. This essentially creates a column stored matrix $[R_g]$. Each row in the matrix corresponds to a u_m point. Each column corresponds to a unique scalar index (u_g point).

The SUPØRT (u_r) and ØMIT (u_o) card images are read from GEØM4. The grid and component numbers are sorted and written on SCR1. The GPDT is read, and a list is formed of the permanent single-point constraints (u_{sg}) which were identified on the GRID bulk data cards. This list is written on SCR1.

The SPCADD, SPC1 and SPC bulk data card images identify the sets of single-point constraints used as boundary points (u_{sb}) and include any possible constrained displacement values (Y_s). The user-requested sets of SPC cards are extracted from the CASECC data block and the SPCADD cards. The requested SPC and SPC1 card images are read, the grid and component "id" numbers are converted to scalar indices, and a paired list of scalar indices and their corresponding displacement values is written on SCR1. The non-zero Y_s values formed in the previous step form a packed vector with indices given by their position in the list. This is the YS data block, a vector packed relative to the u_s set.

The USET data block contains one coded word for each scalar degree of freedom in the entire structure. The word is coded to identify the set or sets of coordinates to which the scalar degree of freedom belongs (see section 2.3.13). An area of core is set equal to

zero, and the lists of u_m, u_o, u_r and u_s points are read from SCR1. The scalar index of a coordinate corresponds to a position in core. The word in that position is modified to identify the set to which that scalar index belongs. With the u_m, u_o, u_r and u_s points known, the u_g, u_n, u_f, u_a and u_ℓ points may be identified. The "nesting" of these sets follows the scheme:



USET is written as one logical record on the USET data block.

The final operation is to process each degree of freedom in USET to insure that the displacement set definitions are consistent. The governing rule (as may be noted in the nesting scheme) is that each degree of freedom may belong to at most one dependent subset. If any inconsistant definitions are found, they are written on SCR1. When each point has been analyzed, EQEXIN and SIL are read into core. Then SCR1 is read, and, for each entry, an error message is queued. PEXIT is then called to abnormally terminate GP4.

4.31.7.1 Definitions of the Sets Defined in USET

- u_g - All structural degrees of freedom defined by grid and scalar points.
- u_m - Dependent coordinates used in the multipoint constraint equations. Defined as the first degree of freedom of a MPC card.
- u_n - All structural degrees of freedom except u_m.
- u_s - All fixed points. The u_{sg} points are defined by the GRID cards and have a displacement of zero. The u_{sb} points are defined by the SPC cards and may have a constrained displacement.

u_f - All degrees of freedom in the structure except u_m and u_s.

u_o - These are "omitted coordinates" defined by ØMIT and ØMIT1 cards. In statics, the structural matrix is partitioned, and these degrees of freedom are solved separately. In dynamics, the displacements of these points are approximated by their static displacements under mass loads.

u_a - These are the unconstrained degrees of freedom of the system. They include rigid body modes in dynamics.

u_r - These are fictitious supports defined by the SUPØRT cards. In dynamics and inertia relief the elastic displacements are measured relative to these points.

u_ℓ - This set includes all degrees of freedom not defined by the u_m, u_s, u_o and u_r points. The stiffness matrix defined by these points is used for the solution of displacements versus loads.

4.31.8 Subroutines

The module GP4 consists of one subroutine, GP4.

4.31.9 Design Requirements

The maximum storage requirement for GP4 is one word per degree of freedom (i.e., LUSET = 6* (number of grid points) + number of scalar points) plus one GINØ buffer. Its open core is defined by /GP4CØR/. Two scratch files are used by GP4.

4.31.10 Diagnostic Messages

The following messages may be issued by GP4:

2047, 2048, 2049, 2050, 2051, 2052, 2053, 2101, 3008

4.32 FUNCTIONAL MODULE GPSP (GRID PØINT SINGULARITY PRØCESSØR).

4.32.1 Entry Point: GPSP.

4.32.2 Purpose

The GPST data block contains data on possible stiffness matrix singularities. These singularities may be removed through the application of single or multipoint constraints. The GPSP module checks each singularity against the list of constraints, and if the singularity is not removed, writes data for warning the user.

4.32.3 DMAP Calling Sequence

```
GPSP    GPL,GPST,USET,SIL/ØGPST  $
```

4.32.4 Input Data Blocks

GPL - Grid Point List.

GPST - Grid Point Singularity Table.

USET - Displacement set definitions table.

SIL - Scalar Index List.

Note: No input data block can be purged.

4.32.5 Output Data Blocks

ØGPST - Unremoved Grid Point Singularities. This data block will be processed by the ØFP (Output File Processor) module.

4.32.6 Parameters

None.

4.32.7 Method

USET is read into core. The USET data block contains one word for each degree of freedom in the structural model. This word identifies the displacement coordinate sets to which the coordinate belongs. Each entry in the GPST data block contains the order of the singularity and the scalar index numbers of the degrees of freedom involved. The logic of the algorithm depends on the order of the singularity. For each type the logic is:

1. If order = 1, the contents of the GPST data are:

    a. ØRDER = 1

b. Number of words following

c. N_1

d. N_2 } may not appear
e. N_3 }

N_1, N_2 and N_3 are the scalar indices of the degrees of freedom which will remove the singularity if constrained. If the singularity is not removed, the ØGPST data is output.

2. If order = 2:

a. ØRDER = 2

b. Number of words following

c. N_{11}

d. N_{21}

e. N_{12} }
f. N_{22} } may not appear
g. N_{13} }
h. N_{23} }

Each pair of indices identifies two degrees of freedom which cause the singularity. If both indices for any pair belong to the USET (UM or US), the singularity is removed. If only one of the degrees of freedom in a pair is constrained by a u_s or u_m coordinate, the singularity order is now ØRDER = 1. The numbers listed are not unique, and more than one of the N_{ij} indices may belong to the u_s or u_m sets. Keeping the same sequence, the unconstrained scalar indices in each partially constrained pair is listed in the ØGPST in the form for ØRDER = 1.

3. If order = 3:

a. ØRDER = 3

b. Number of words following

c. N_1

d. N_2

e. N_3

All three indices (N_1, N_2, and N_3) must belong to the u_s or u_m sets to remove the

singularity. If one or two of the coordinates are constrained, the order is two or one, and the remaining scalar indices or index is listed in the ØGPST.

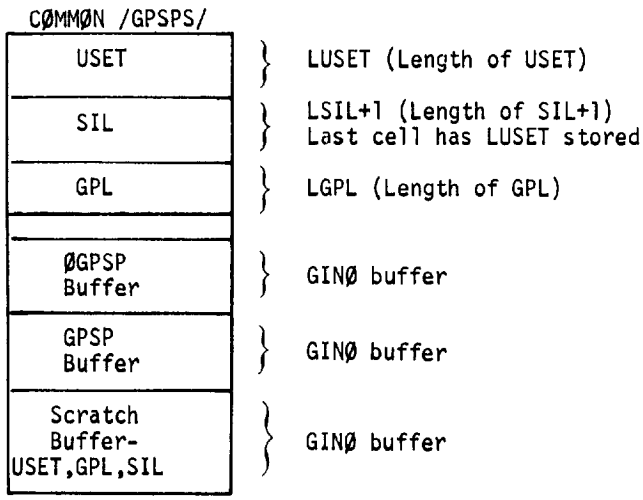
If any singularities are unremoved, a message is given to this effect. Also the SIL values must be converted to external grid point-component notation for user readability. The first time an unremoved singularity is detected, SIL is read into core beneath USET, and GPL is placed beneath SIL. Each SIL value is looked up in SIL for a pointer into GPL to the external grid point ID. Scalar points are differentiated from grid points. Data is output to the ØGPST in the order: grid point ID, type, singularity order and components. When all entries in the GPST have been processed, the routine returns.

4.32.8 Subroutines

None.

4.32.9 Design Requirements

Open core is defined at /GPSPA/.



4.32.9.1 Allocation of Core Storage

If no singularities exist, USET plus two GINØ buffers must be held in core. If singularities exist, USET, SIL, GPL plus three GINØ buffers must be held in core.

4.32.10 Diagnostic Messages

The following diagnostic messages may occur: 3007 (if the GPST does not contain legal SIL numbers, indicating a programming error); 3008 (if the core storage requirements given in section 4.32.9.1 are not met).

4.33 FUNCTIONAL MODULE MCE1 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 1)

4.33.1 Entry Point: MCE1

4.33.2 Purpose

MCE1 partitions $[R_g]$ into $[R_m]$ and $[R_n]$ and then solves the matrix equation $[R_m]\,[G_m] = -[R_n]$ for $[G_m]$.

4.33.3 DMAP Calling Sequence

```
MCE1    USET,RG/GM  $
```

4.33.4 Input Data Blocks

USET - Displacement set definitions table.

RG - Multipoint constraint equations matrix.

Note: Neither USET nor RG may be purged.

4.33.5 Output Data Blocks

GM - Multipoint constraint transformation matrix - m set.

Note: GM may not be purged.

4.33.6 Parameters

None

4.33.7 Method

$[R_g]$ is a matrix with each row defining a constraint equation. The row scalar indices correspond to the u_m set of coordinates and the column indices correspond to the u_g set. The first operation of MCE1 is to partition $[R_g]$ into $[R_m]$ and $[R_n]$. MCE1A performs this operation by initializing /PARMEG/ and calling PARTN (see section 3.5.6 for PARTN details).

The second operation of MCE1 is to solve the matrix equation

$$[R_m]\,[G_m] = -[R_n] \qquad (1)$$

for $[G_m]$. If $[R_m]$ is diagonal, the operation is straightforward. In this case MCE1D is called. The terms $-r_{ii}$ are stored in core, the $[R_n]$ matrix is read interpretively by INTPK, and the terms of $[G_m]$ are formed from the equation

$$g_{ij} = - \frac{r_{ij}}{r_{ii}}, \qquad (2)$$

where the terms in the numerator belong to $[R_n]$ and those in the denominator belong to $[R_m]$. If $[R_m]$ is not diagonal, Equation 1 is solved by decomposition and forward-backward substitution. In this case, MCE1B is called. MCE1B performs an unsymmetric decomposition of $[R_m]$ by initializing /DCØMPX/ and calling DECØMP. MCE1C is then called by MCE1. MCE1C performs a forward-backward substitution to solve for $[G_m]$ by initializing /GFBSX/ and calling GFBS. See section 3.5.15 and 3.5.19 for further details on DECØMP and GFBS, respectively.

4.33.8 Subroutines

4.33.8.1 Subroutine Name: MCE1A

1. Entry Point: MCE1A
2. Purpose: To partition $[R_g]$ into $[R_m]$ and $[R_n]$.
3. Calling Sequence: CALL MCE1A

4.33.8.2 Subroutine Name: MCE1B

1. Entry Point: MCE1B
2. Purpose: To decompose $[R_m]$ into lower and upper triangular factors.
3. Calling Sequence: CALL MCE1B

4.33.8.3 Subroutine Name: MCE1D

1. Entry Point: MCE1D
2. Purpose: To solve the matrix equation $[R_m]\,[G_m] = [R_n]$ for $[G_m]$ where $[R_m]$ is diagonal.
3. Calling Sequence: CALL MCE1D

4.33.9 Design Requirements

4.33.9.1 Allocation of Core Storage

The maximum core storage requirement in the module is one double precision vector in the u_m displacement set plus three GINØ buffers.

4.33.9.2 Environment

Communication of GINØ file names to each of the phases of MCE1 occurs through blank CØMMØN. The four phases are designed so that each may be in a separate overlay segment. Open core for each of the phases is as follows:

MCE1A: /MCEA1/

MCE1B: /MCEB1/

MCE1C: /MCEC1/

MCE1D: /MCED1/.

4.33.10 Diagnostic Messages

The following messages may be issued by MCE1:

3005, 3016

4.34 FUNCTIONAL MODULE MCE2 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 2)

4.34.1 Entry Point: MCE2

4.34.2 Purpose

MCE2 partitions the stiffness matrix $[K_{gg}]$ into $[\overline{K}_{nn}]$, $[K_{mn}]$ and $[\overline{K}_{mm}]$ and then performs the matrix reduction $[K_{nn}] = [\overline{K}_{nn}] + [G_m]^T [\overline{K}_{mn}] + [\overline{K}_{mn}]^T [G_m] + [G_m]^T [\overline{K}_{mm}] [G_m]$

Similar partitions and reductions are performed on $[M_{gg}]$, $[B_{gg}]$ and $[K^4_{gg}]$ if these matrices are not purged.

4.34.3 DMAP Calling Sequence

```
MCE2    USET,GM,KGG,MGG,BGG,K4GG/KNN,MNN,BNN,K4NN  $
```

4.34.4 Input Data Blocks

USET - Displacement set definitions table.

GM - Multipoint constraint transformation matrix - m set.

KGG - Partition of stiffness matrix - g set.

MGG - Partition of mass matrix - g set.

BGG - Partition of damping matrix - g set.

K4GG - Partition of structural damping matrix - g set.

Note: MGG, BGG and K4GG may be purged.

4.34.5 Output Data Blocks

KNN - Partition of stiffness matrix - n set.

MNN - Partition of mass matrix - n set.

BNN - Partition of damping matrix - n set.

K4NN - Partition of structural damping matrix - n set.

Note: MNN, BNN or K4NN may be purged only if MGG, BGG or K4GG is purged.

4.34.6 Parameters

None

4.34.7 Method

Using subroutine UPART to generate row and column partitioning vectors and subroutine MPART to perform the actual partitioning, $[K_{gg}]$ is partitioned as follows:

$$[K_{gg}] \Longrightarrow \left[\begin{array}{c|c} \overline{K}_{nn} & \overline{K}_{nm} \\ \hline \overline{K}_{mn} & \overline{K}_{mm} \end{array}\right]. \tag{1}$$

Subroutine ELIM is called to perform the following matrix reduction:

$$[K_{nn}] = [\overline{K}_{nn}] + [G_m]^T [\overline{K}_{mn}] + [\overline{K}_{mn}]^T [G_m] + [G_m]^T [\overline{K}_{mm}] [G_m]. \tag{2}$$

For each of the data blocks corresponding to the matrices $[M_{gg}]$, $[B_{gg}]$, $[K^4_{gg}]$ which is not purged, the above partitioning and matrix reductions are performed.

4.34.8 Subroutines

Calls are made to the following matrix utility routines:

UPART see section 3.5.9 for details

MPART see section 3.5.9 for details

ELIM see section 3.5.22 for details

4.34.9 Design Requirements

4.34.9.1 Allocation of Core Storage

The maximum storage requirement for MCE2 is four times the number of degrees of freedom in the u_n displacement set plus one GINØ buffer.

4.34.9.2 Environment

The module MCE2 consists of one subroutine, MCE2. Calls are made to the matrix utility routines as indicated above. Six scratch files are used.

4.35 FUNCTIONAL MODULE SCE1 (SINGLE-POINT CONSTRAINT ELIMINATOR)

4.35.1 Entry Point: SCE1

4.35.2 Purpose

To reduce the n set matrices to f set matrices by removing the single-point constraints.

4.35.3 DMAP Calling Sequence

```
SCE1  USET,KNN,MNN,BNN,K4NN/KFF,KFS,KSS,MFF,BFF,K4FF/  $
```

4.35.4 Input Data Blocks

USET - Displacement set definitions table.

KNN - Partition of stiffness matrix - n set.

MNN - Partition of mass matrix - n set.

BNN - Partition of damping matrix - n set.

K4NN - Partition of the structural damping matrix - n set.

Notes: 1. USET cannot be purged.

2. KNN,MNN,BNN and K4NN can be purged.

3. At least one degree of freedom must belong to the f and s sets.

4.35.5 Output Data Blocks

KFF - Partition of stiffness matrix after single-point constraints have been removed - f set.

KFS - Partition of stiffness matrix after single-point constraints have been removed.

KSS - Partition of stiffness matrix after single-point constraints have been removed - s set.

MFF - Partition of mass matrix after single-point constraints have been removed - s set.

BFF - Partition of damping matrix after single-point constraints have been removed - f set.

K4FF - Partition of structural damping matrix with single-point constraints removed - f set.

4.35.6 Parameters

None

4.35.7 Method

The matrices are partitioned using USET(UN,UF,US) as follows (see section 1.7 for details):

1. If $[K_{nn}]$ exists:

$$[K_{nn}] \Rightarrow \left[\begin{array}{c|c} K_{ff} & K_{fs} \\ \hline K_{sf} & K_{ss} \end{array}\right] . \qquad (1)$$

The $[K_{ff}]$, $[K_{fs}]$ and $[K_{ss}]$ partitions are generated and saved.

2. If $[M_{nn}]$ exists:

$$[M_{nn}] \Rightarrow \left[\begin{array}{c|c} M_{ff} & M_{fs} \\ \hline M_{sf} & M_{ss} \end{array}\right] . \qquad (2)$$

3. If $[B_{nn}]$ exists:

$$[B_{nn}] \Rightarrow \left[\begin{array}{c|c} B_{ff} & B_{fs} \\ \hline B_{sf} & B_{ss} \end{array}\right] . \qquad (3)$$

4. If $[K^4_{nn}]$ exists:

$$[K^4_{nn}] \Rightarrow \left[\begin{array}{c|c} K^4_{ff} & K^4_{fs} \\ \hline K^4_{sf} & K^4_{ss} \end{array}\right] . \qquad (4)$$

For the $[M_{nn}]$, $[B_{nn}]$ and $[K^4_{nn}]$ matrices, only the "ff" partition is generated and saved.

One call to UPART followed by 4 calls to MPART accomplishes the above tasks.

4.35.8 Subroutines

UPART and MPART are called. See subroutine description in section 3.5.9.

4.35.9 Design Requirements

One scratch file is necessary.

4.36 FUNCTIONAL MODULE SMP1 (STRUCTURAL MATRIX PARTITIONER - PHASE 1)

4.36.1 Entry Point: SMP1

4.36.2 Purpose

SMP1 partitions $[K_{ff}]$ into $[\bar{K}_{aa}]$, $[\bar{K}_{oa}]$ and $[\bar{K}_{oo}]$. The matrix equation $[\bar{K}_{oo}]\,[G_o] = -[\bar{K}_{oa}]$ is solved for $[G_o]$. $[K_{ff}]$ is then reduced by the matrix equation $[K_{aa}] = [\bar{K}_{aa}] + [\bar{K}_{oa}]^T\,[G_o]$. If $[M_{ff}]$ is not purged, it is reduced by the equation $[M_{aa}] = [\bar{M}_{aa}] + [G_o]^T\,[\bar{M}_{oa}] + [\bar{M}_{oa}]^T[G_o] + [G_o]^T\,[\bar{M}_{oo}]\,[G_o]$. Similarly, $[B_{ff}]$ and $[K^4_{ff}]$ are reduced.

4.36.3 DMAP Calling Sequence

```
SMP1    USET,KFF,MFF,BFF,K4FF/GØ,KAA,KØØB,LØØ,UØØ,MAA,MØØB,MØAB,BAA,K4AA  $
```

4.36.4 Input Data Blocks

USET - Displacement set definitions table.

KFF - Partition of stiffness matrix - f set.

MFF - Partition of mass matrix - f set.

BFF - Partition of damping matrix - f set.

K4FF - Partition of structural damping matrix - f set.

Note: MFF, BFF or K4FF may be purged.

4.36.5 Output Data Blocks

GØ - Structural matrix partitioning transformation matrix.

KAA - Partition of stiffness matrix - a set.

KØØB - Partition of stiffness matrix - o set.

LØØ - Lower triangular factor of KØØB - o set.

UØØ - Upper triangular factor of KØØB - o set.

MAA - Partition of mass matrix - a set.

MØØB - Partition of mass matrix - a set.

MØAB - Partition of mass matrix.

BAA - Partition of damping matrix - a set.

K4AA - Partition of structural damping matrix - a set.

Note:

1. UØØ is not a standard upper triangular factor. Its format is compatible only for input to subroutine FBS.

2. MAA, MØØB, MØAB, BAA or K4AA may be purged only if MFF, BFF or K4FF are purged.

4.36.6 Parameters

None

4.36.7 Method

Using subroutine UPART to generate row and column partitioning vectors and subroutine MPART to perform the actual partitioning, $[K_{ff}]$ is partitioned as follows:

$$[K_{ff}] \Rightarrow \left[\begin{array}{c|c} \bar{K}_{aa} & \bar{K}_{ao} \\ \hline \bar{K}_{oa} & \bar{K}_{oo} \end{array}\right]. \tag{1}$$

Subroutine FACTØR is called to decompose $[\bar{K}_{oo}]$ into triangular factors. Subroutine SØLVER is called to perform a forward-backward substitution solving for $[G_o]$ in the matrix equation

$$[\bar{K}_{oo}]\,[G_o] = -[\bar{K}_{oa}]\,, \tag{2}$$

and computing $[K_{aa}]$ from the equation

$$[K_{aa}] = [\bar{K}_{aa}] + [\bar{K}_{oa}]\,[G_o]. \tag{3}$$

For each of the data blocks $[M_{ff}]$, $[B_{ff}]$, $[K^4_{ff}]$ which is not purged, the above partitioning operation is performed and the matrix reductions:

$$[M_{aa}] = [\bar{M}_{aa}] + [G_o]^T\,[\bar{M}_{oa}] + [\bar{M}_{oo}]^T\,[G_o] + [G_o]^T\,[\bar{M}_{oo}]\,[G_o], \tag{4}$$

$$[B_{aa}] = [\bar{B}_{aa}] + [G_o]^T\,[\bar{B}_{oa}] + [\bar{B}_{oa}]^T\,[G_o] + [G_o]^T\,[\bar{B}_{oo}]\,[G_o], \tag{5}$$

$$[K^4_{aa}] = [\bar{K}^4_{aa}] + [G_o]^T\,[\bar{K}^4_{oa}] + [\bar{K}^4_{oa}]^T\,[G_o] + [G_o]^T\,[\bar{K}^4_{oo}]\,[G_o], \tag{6}$$

are performed by subroutine ELIM.

4.36.8 Subroutines

Calls are made to the following matrix utility routines:

UPART See subroutine descriptions - section 3.5.9 for details

MPART See subroutine descriptions - section 3.5.9 for details

FACTØR See subroutine descriptions - section 3.5.23 for details

SØLVER See subroutine descriptions - section 3.5.20 for details

ELIM See subroutine descriptions - section 3.5.22 for details

4.36.9 Design Requirements

4.36.9.1 Allocation of Core Storage

The maximum storage requirement for SMP1 is four times the number of degrees of freedom in the u_a displacement set plus one GINØ buffer.

4.36.9.2 Environment

The module SMP1 consists of one subroutine, SMP1. Calls are made to the matrix utility routines indicated above. Six scratch files are used.

4.37 FUNCTIONAL MODULE RBMG1 (RIGID BODY MATRIX GENERATOR - PHASE 1)

4.37.1 Entry Point: RBMG1

4.37.2 Purpose

RBMG1 partitions $[K_{aa}]$ into $[K_{\ell\ell}]$, $[K_{\ell r}]$ and $[K_{rr}]$. If $[M_{aa}]$ is not purged, it is partitioned similarly.

4.37.3 DMAP Calling Sequence

```
RBMG1     USET,KAA,MAA/KLL,KLR,KRR,MLL,MLR,MRR  $
```

4.37.4 Input Data Blocks

USET - Displacement set definitions table.

KAA - Partition of stiffness matrix - a set.

MAA - Partition of mass matrix - a set.

Note: USET may not be purged.

4.37.5 Output Data Blocks

KLL - Partition of stiffness matrix - ℓ set.

KLR - Partition of stiffness matrix.

KRR - Partition of stiffness matrix - r set.

MLL - Partition of mass matrix - ℓ set.

MLR - Partition of mass matrix.

MRR - Partition of mass matrix - r set.

Note: Output data blocks may be purged only if the corresponding input data block is purged.

4.37.6 Parameters

None

4.37.7 Method

Using subroutine UPART to generate row and column partitioning vectors and subroutine

MPART to perform the actual partitioning, $[K_{aa}]$ is partitioned as follows:

$$[K_{aa}] \Longrightarrow \left[\begin{array}{c|c} K_{\ell\ell} & K_{\ell r} \\ \hline K_{r\ell} & K_{rr} \end{array}\right]. \qquad (1)$$

Similarly, if $[M_{aa}]$ is not purged, it is partitioned.

4.37.8 Subroutines

RBMG1 calls the following matrix utility routines:

UPART, MPART } see section 3.5.9 for details.

4.37.9 Design Requirements

4.37.9.1 Allocation of Core Storage

Storage requirements for RBMG1 are minimal since no unpacked vectors are held in core.

4.37.9.2 Environment

The module RBMG1 consists of one subroutine, RBMG1. Calls are made to the matrix utility routines indicated above. One scratch file is used.

4.38 FUNCTIONAL MODULE RBMG2 (RIGID BODY MATRIX GENERATOR - PHASE 2)

4.38.1 Entry Point: RBMG2

4.38.2 Purpose

RBMG2 decomposes $[K_{\ell\ell}]$ into its triangular factors $[L_{\ell\ell}]$ and $[U_{\ell\ell}]$.

4.38.3 DMAP Calling Sequence

```
RBMG2    KLL/LLL,ULL/V,N,PØWER/V,N,DET  $
```

4.38.4 Input Data Blocks

KLL - Partition of stiffness matrix - ℓ set.

Note: KLL may not be purged.

4.38.5 Output Data Blocks

LLL - Lower triangular factor of KLL - ℓ set.

ULL - Upper triangular factor of KLL - ℓ set.

Notes

1. LLL and ULL may not be purged.
2. ULL is not a standard upper triangular matrix. Its format is compatible only for input to subroutine FBS.

4.38.6 Parameters

PØWER - Output-integer-default = 1. Power of 10 in the determinant of KLL.

DET - Output-real-default = 1.0. Magnitude of determinant of KLL, i.e., $\det [K_{\ell\ell}] = \text{DET} * 10^{\text{PØWER}}$

4.38.7 Method

RBMG2 calls subroutine FACTØR to perform the decomposition of $[K_{\ell\ell}]$ into $[L_{\ell\ell}]$ and $[U_{\ell\ell}]$.

4.38.8 Subroutines

RBMG2 calls the matrix utility routine FACTØR (see section 3.5.23 for FACTØR details).

4.38.9 Design Requirements

For allocation of core storage, see subroutine SDCØMP (section 3.5.14). The module RBMG2 consists of one subroutine, RBMG2. Three scratch files are used.

4.38.10 Diagnostic Messages

Message number 3005 may be issued by RBMG1.

4.39 FUNCTIONAL MODULE RBMG3 (RIGID BODY MATRIX GENERATOR - PHASE 3)

4.39.1 Entry Point: RBMG3

4.39.2 Purpose

RBMG3 solves for the rigid body transformation matrix [D] from the equation

$$[K_{\ell\ell}]\,[D] = -[K_{\ell r}] \quad . \tag{1}$$

The rigid body error ratio, ε, is computed from

$$\varepsilon = \frac{||[K_{rr}] + [K_{\ell r}]^T\,[D]||}{||[K_{rr}]||} \quad . \tag{2}$$

4.39.3 DMAP Calling Sequence

```
RBMG3     LLL,ULL,KLR,KRR/DM     $
```

4.39.4 Input Data Blocks

LLL - Lower triangular factor of KLL -ℓ set.

ULL - Upper triangular factor of KLL - ℓ set.

KLR - Partition of stiffness matrix.

KRR - Partition of stiffness matrix - r set.

Note: Input data blocks may not be purged.

4.39.5 Output Data Blocks

DM - Rigid body transformation matrix.

Note: The DM data block corresponds to the matrix [D] and may not be purged.

4.39.6 Parameters

None

4.39.7 Method

Subroutine SØLVER is called to perform the operations in Equation 1 and 2.

4.39.8 Subroutines

RBMG3 calls the matrix utility routine SØLVER and has no auxiliary subroutines. See section 3.5.20 for SØLVER details.

4.39.9 Design Requirements

For allocation of core storage, see subroutines FBS (section 3.5.17) and MPYAD (section 3.5.12). Two scratch files are used.

4.40 FUNCTIONAL MODULE RBMG4 (RIGID BODY MATRIX GENERATOR - PHASE 4)

4.40.1 Entry Point: RBMG4

4.40.2 Purpose

RBMG4 computes the rigid body mass matrix $[m_r]$ from the matrix equation

$$[m_r] = [M_{rr}] + [D]^T [M_{\ell r}] + [M_{\ell r}]^T [D] + [D]^T [M_{\ell\ell}] [D]. \qquad (1)$$

4.40.3 DMAP Calling Sequence

```
RBMG4     DM,MLL,MLR,MRR/MR  $
```

4.40.4 Input Data Blocks

DM - Rigid body transformation matrix.

MLL - Partition of mass matrix - ℓ set.

MLR - Partition of mass matrix.

MRR - Partition of mass matrix - r set.

Notes:

1. No input data block may be purged.
2. The DM data block corresponds to the matrix [D] in Equation 1.

4.40.5 Output Data Blocks

MR - Rigid body mass matrix - r set.

4.40.6 Parameters

None

4.40.7 Method

Subroutine ELIM is called to compute $[m_r]$ as in Equation 1 (see section 3.5.22 for ELIM details).

4.40.8 Subroutines

RBMG4 consists of one subroutine, RBMG4.

Matrix utility routine ELIM (section 3.5.22) is called by RBMG4.

4.40.9 Design Requirements

For allocation of core storage, see subroutine MPYAD (section 3.5.12). Three scratch files are used.

4.41 FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

4.41.1 Entry Point: SSG1

4.41.2 Purpose

To compute the static loads, thermal loads, and enforced deformation loads selected by the user.

4.41.3 DMAP Calling Sequence

```
SSG1    SLT,BGPDT,CSTM,SIL,EST,MPT,GPTT,EDT,MGG,CASECC,DIT/PG/V,N,LUSET/V,N,NSKIP   $
```

4.41.4 Input Data Blocks

SLT - Static Loads Table.

BGPDT - Basic Grid Point Definition Table.

CSTM - Coordinate System Transformation Matrices.

SIL - Scalar Index List.

EST - Element Summary Table.

MPT - Material Property Table.

GPTT - Grid Point Temperature Table.

EDT - Element Deformation Table.

MGG - Partition of mass matrix - g set.

CASECC - Case Control Data Table.

DIT - Direct Input Tables.

Notes:

1. SLT, BGPDT, SIL cannot be purged if external static loads or LØAD cards are selected in CASECC.

2. CSTM cannot be purged if any grid point or load references a coordinate system other than basic.

3. EST, MPT cannot be purged if thermal or element deformation loads are selected.

4. GPTT cannot be purged if thermal loads are applied.

5. EDT cannot be purged if element deformation loads are selected.

6. MGG cannot be purged if GRAVITY or RFØRCE loads are applied.

7. CASECC cannot be purged.

8. DIT cannot be purged if temperature dependent materials are loaded.

4.41.5 Output Data Blocks

PG - Static load vector matrix giving static loads - g set.

Note: PG can never be purged.

4.41.6 Parameters

LUSET - Input-integer-no default. LUSET defines length of PG.

NSKIP - Input-integer-no default. One static load is built for each CASECC record starting with NSKIP + 1 as long as the boundary conditions are constant.

4.41.7 Overview of the Method Used in SSG1

The purpose of the first phase of static solution calculation (module SSG1) is the generation of the load vectors on the whole structure. The structure may be loaded in three different ways:

1. Simple applied loads and moments may be given to grid and scalar points. Pressure loads may be applied to an area defined by three or four grid points. Centrifugal force fields may also be defined.

2. Thermal and enforced deformation loads are generated by using the structural element characteristics. The loads on the connected grid points are equivalent to fixing the displacements and replacing the element by the load it would apply to the points.

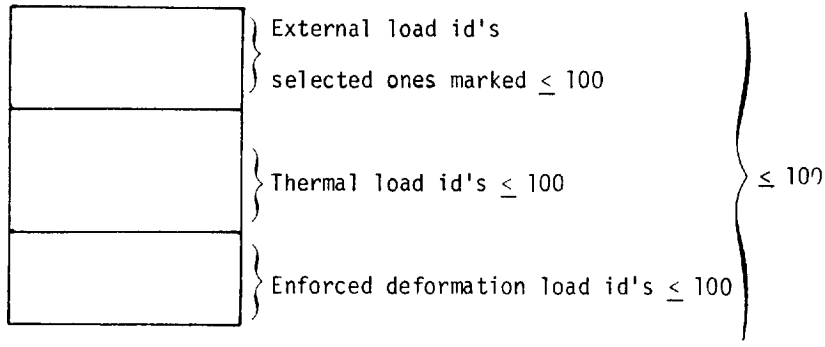
3. Gravity loads are dependent on the mass characteristics of the structure. A gravity load is produced by generating a vector of accelerations on all grid points in the structure and pre-multiplying the vector by the structural mass matrix.

The details on load vector generation for these three different types of loading are given in sections 4.41.8, 4.41.9 and 4.41.10 below. The function of module SSG1 is to read the case control data and extract the necessary load set data, calculate load vectors for each requested load set, and combine these sets to produce the loads requested for all subcases using the same boundary condition.

4.41.7.1 Module Initialization

Common block /LØADX/, which contains GINØ file numbers for input data files and position pointers, is initialized.

A list of all external load sets is extracted from the SLT. (This must be less than 101.) CASECC is skipped forward NSKIP records (in case several boundary conditions are being solved in one run). For each succeeding record which is not an eigenvalue record, not a symmetry record, not a differential stiffness record, and for which the boundary conditions are those for the current loop (SPC and MPC sets), a list is made of each thermal or enforced deformation load. The external loads selected are marked in the above list. If a selected external load is not in the above list, the LØAD cards are read in and their component id's searched. A LØAD card may cause additional members to be selected. A composite list is created which contains:



If there is no record which allows construction of a load, SSG1 aborts. If a selected external load id does not exist either as a LØAD card or simple load set, SSG1 aborts. (Subroutine SSG1A).

4.41.7.2 Individual Load Vector Generation

Each requested set of loads is used to generate a $\{P_g^j\}$ load vector. The vectors are generated one at a time in core and written on the PG temporary file, a scratch file. Files PG temporary, SLT, BGPDT, CSTM and SIL are opened. The vector generation depends on the type of load and type of input data. Details are given in sections 4.41.8, 4.41.9 and 4.41.10 below.

4.41.7.3 Subcase Load Vector Generation

Each simple load set, j, produces a $\{P_g^j\}$ load vector, and each subcase may be a combination of various simple load sets. As each load set vector is formed, it is written on PG temporary. When all sets have been generated, the CASECC data block and the LØAD card images are read again. A table is formed for each subcase consisting of the required set number and the scale factor for each set if given on a LØAD card. The file containing the load vectors for the sets is read for each subcase, c, and added to a $\{P_g^c\}$ load vector. The $\{P_g^c\}$ load vectors are packed and written as the PG data block in standard NASTRAN matrix form.

4.41.8 Direct Applied Loads

Direct loads are applied to the structural model by means of FØRCE, FØRCE1, FØRCE2, GRAV, MØMENT, MØMENT1, MØMENT2, PLØAD, PLØAD2, RFØRCE, and SLØAD Bulk Data Cards and the PRESAX card which is used for the axisymmetric conical shell problem only.

4.41.8.1 FØRCE and MØMENT Card Processing

The data described by a FØRCE or MØMENT data card are given as follows:

N_p = Grid point index;

N_c = Coordinate system number;

S = Scalar factor; and

$$\{P\} = \begin{Bmatrix} A_1 \\ A_2 \\ A_3 \end{Bmatrix} . \qquad (1)$$

The BGPDT data for the point are determined. If the global coordinate number, N_g, for the point equals N_c, the vector is

$$\{P_g\} = S\,\{P\} , \qquad (2)$$

where the row index is determined from SIL. If $N_g \neq N_c$, $[T_g]$ and $[T_c]$ are calculated using the location coordinates and the two local coordinate systems (subroutines GLBBS and BASGLB). The

load in global coordinates is:

$$\{P_g\} = S\,[T_g]^T[T_c]\,\{P\}\ . \tag{3}$$

If a FØRCE card is used, the loads are added to the first three positions for the grid point in the load vector. If a MØMENT card is used, the loads are added to the last three positions. (Subroutine DIRECT).

4.41.8.2 FØRCE1 and MØMENT1 Card Processing

The data described by FØRCE1 or MØMENT1 card are given as follows:

N_p = Application point number;

S = Load magnitude;

N_1, N_2 = Grid point numbers describing the vector direction of the load.

The basic coordinates of the points N_1, N_2 and N_p are found in the BGPDT. (Subroutines PERMUT and FNDPNT). If $\{R_1\}$ and $\{R_2\}$ are the vectors corresponding to N_1 and N_2, the load direction is:

$$\{d\} = \frac{\{R_2\} - \{R_1\}}{|\{R_2\} - \{R_1\}|}. \tag{4}$$

The coordinate transformation $[T_g]$ for point N_p is calculated (Subroutine BASGLB). The load vector in global coordinates is:

$$\{P_q\} = S\,[T_g]^T\{d\}\ . \tag{5}$$

If a FØRCE1 card was used the values are added to the first three coordinates, starting with the SIL number, in the load vector. If a MØMENT1 card was used, the values are added to the last three (subroutine TPØNT).

4.41.8.3 FØRCE2 and MØMENT2 Card Processing

The data on a FØRCE2 or MØMENT2 card are as follows:

N_p = Application point number;

S = Load magnitude;

N_1, N_2, N_3, N_4 are such that the direction of the force is determined by Equation 6 below.

The algorithm is similar to the one for the FØRCE1 and MØMENT1 case except that four basic coordinate system vectors, $\{R_1\}$, $\{R_2\}$, $\{R_3\}$, $\{R_4\}$, are formed for the four points and:

$$\{d\} = \frac{(\{R_2\} - \{R_1\}) \times (\{R_4\} - \{R_3\})}{|(\{R_2\} - \{R_1\}) \times (\{R_4\} - \{R_3\})|} , \tag{6}$$

(Subroutine FPØNT).

4.41.8.4 PLØAD and PLØAD2 Card Processing

The data contents for a PLØAD card are (a PLØAD2 card is transformed into a PLØAD card by GP3):

p	=	Pressure value;
N_1, N_2, N_3, N_4	=	Points describing area over which pressure load is acting. (N_4 is optional.)

For each of the four points, N_i, the basic coordinate system vector, $\{R_i\}$, is formed.

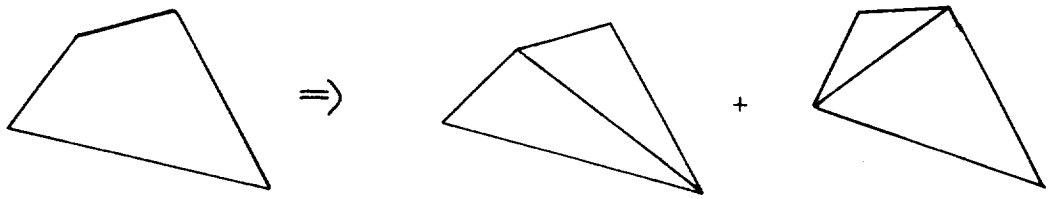
If $N_4 = 0$, the load on each point is:

$$\{F\} = -\frac{p}{6}\left[(\{R_1\} - \{R_2\}) \times (\{R_3\} - \{R_2\})\right] . \tag{7}$$

The load vector in global coordinates for each point is:

$$\{P_{g_i}\} = [T_i]^T\{F\} . \tag{8}$$

If $N_4 \neq 0$, the quadrilateral is subdivided into four triangles as shown.



For each triangle $\{P_{g_i}\}$ is calculated for the three connected points using a pressure value $p_T = 1/2\ p$. The equations are the same as the previous case except that the points are interchanged each time the triangle calculation is done. (Subroutine PLØAD.)

4.41.8.5 SLØAD Card Processing

The data contents for a SLØAD card are:

N_p = Scalar point id and

S = Load on point.

The scalar index is computed by subroutine FNDSIL, and S is added in. (Subroutine SLØAD.)

4.41.8.6 RFØRCE Card Processing

The data contents for an RFØRCE card are:

N_p = Index of grid point through which rotation vector passes;

N_c = Coordinate system number defining the rotation vector;

A = Factor for vector;

R_x, R_y, R_z are components of rotation vector in cps.

The following sequence of operations comprises RFØRCE card processing which is carried out in subroutine RFØRCE.

1. The local to basic coordinate transformation matrix, $[T_c]$, for the reference coordinate system N_c is extracted from the CSTM data block.

2. The rotation vector in basic coordinates, and in radians per second, is:

$$\{\omega_b\} = 2\pi A [T_c] \begin{Bmatrix} R_x \\ R_y \\ R_z \end{Bmatrix} \tag{9}$$

3. Define the basic location vector of the reference point N_p as $\{r_p\}$. If $N_p = 0$, set $\{r_p\} = \{0\}$.

4. Extract the basic coordinate system vector $\{r_i\}$ for each point from BGPDT.

5. Using $\{r_i\}$ and the local coordinate system referenced by the point, calculate the global-to-basic transformation matrix $[T_i]$.

6. For the six columns of the mass matrix $[M_{gg}]$ corresponding to the grid point i, the 6x6 matrix partition on the diagonal is extracted. Define this as $[M^i]$.

7. Partition the 6x6 matrix into 3x3 matrices

$$[M^i] \Longrightarrow \left[\begin{array}{c|c} M_t^i & M_{tr}^i \\ \hline M_{rt}^i & M_r^i \end{array}\right] , \tag{10}$$

and transform the rotational velocity vector to global coordinates

$$\{\omega_g\} = [T_i]^T \{\omega_b\} . \tag{11}$$

8. Calculate the forces and moments on the grid point by the equations:

$$\{F\} = -\{\omega_g\} \times [M_t^i] [T_i]^T(\{\omega_b\} \times [\{r_i\} - \{r_a\}]) - \{\omega_g\} \times [M_{tr}^i] \{\omega_g\} \tag{12}$$

$$\{M\} = -\{\omega_g\} \times [M_{rt}^i] [T_i]^T (\{\omega_b\} \times [\{r_i\} - \{r_a\}]) - \{\omega_g\} \times [M_r^i] \{\omega_g\} \tag{13}$$

9. The load vector partition in global coordinates is:

$$\{P_i\} = \left\{\begin{array}{c} F \\ \hline M \end{array}\right\} \tag{14}$$

4.41.8.7 PRESAX Card Processing

The data contents for a PRESAX card (which applies only to pressure loading of an AXISYMMETRIC shell) are:

P = Pressure;

N_a = Index value of harmonic Ring A;

N_b = Index value of harmonic Ring B;

ϕ_1 = (degrees);

ϕ_2 = (degrees);

n = Harmonic number of harmonic being added.

The algorithm given in the following steps is performed in subroutine PRESAX.

1. ϕ_1 and ϕ_2 are converted to radians.

2. BGPDT data are extracted for both RINGA and RINGB, giving {r} and {z}.

3. The calculations for harmonic n are:

$$\ell = \sqrt{(r_b-r_a)^2 + (z_b-z_a)^2}, \tag{15}$$

$$\sin \psi = \frac{r_b-r_a}{\ell}, \tag{16}$$

$$\cos \psi = \frac{z_b-z_a}{\ell}. \tag{17}$$

For the cosine case, if n = 0, we calculate:

$$P^i_{ro} = P\ell \left(\frac{r_i}{3} + \frac{r_j}{6}\right)(\phi_2 - \phi_1) \cos \psi, \tag{18}$$

$$P^i_{zo} = -P\ell \left(\frac{r_i}{3}+\frac{r_j}{6}\right)(\phi_2 - \phi_1) \sin \psi. \tag{19}$$

If n > 0

$$P^i_{rn} = P \frac{\ell}{n} \left(\frac{r_i}{3} + \frac{r_j}{6}\right) (\sin (n\phi_2) - \sin (n\phi_1)) \cos \psi, \tag{20}$$

$$P^i_{zn} = -P \frac{\ell}{n} \left(\frac{r_i}{3} + \frac{r_j}{6}\right) (\sin (n\phi_2) - \sin (n\phi_1)) \sin \psi. \tag{21}$$

For the sine case, if n > 0,

$$P^i_{rn} = - P \frac{\ell}{n} \left(\frac{r_i}{3} + \frac{r_j}{6} \right) (\cos (n\phi_2) - \cos (n\phi_1)) \cos \psi, \tag{22}$$

$$P^i_{zn} = P \frac{\ell}{n} \left(\frac{r_i}{3} + \frac{r_j}{6} \right) (\cos (n\phi_2) - \cos (n\phi_1)) \sin \psi. \tag{23}$$

4. The above equations are solved for i = a, j = b and i = b, j = a. The loads are added to the corresponding grid point location in the PG load vector

$$\left\{ P^i_g \right\} = \left\{ P^i_g \right\} + \begin{Bmatrix} P^i_r \\ 0_i \\ P_z \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{24}$$

(Subroutine PRESAX).

4.41.8.8 GRAV Card Processing

Each GRAV input card describes a uniform acceleration field with the following parameters:

N = Coordinate system ID;

G = Scale factor;

{V} = Vector of load in coordinate system N.

The gravity vector in basic coordinates is:

$$\{g_b\} = G [T_{0N}] \{V\} \tag{25}$$

where $[T_{0N}]$ is the 3x3 orientation matrix of coordinate system N. (Subroutines GRAV, FDCSTM, MPYL). This vector $\{g_b\}$ is saved for later processing. Subroutine EXTERN then returns, noting the number of gravity loads listed.

4.41.9 Thermal and Enforced Deformation Loads

The thermal and enforced deformation loads are calculated using the stiffness properties of the structural elements. The GPTT or EDT data for each load set and the MPT, DIT and SIL data blocks are placed in core. The EST data block is read one element at a time, and the loads produced by that element are placed in the PG load vector. The actual algorithms for generating element loads are given in section 4.87.

4.41.10 Gravity Loads

Acceleration vectors are computed for each gravity load by two means, one for an axisymmetric shell problem, the other for non-shell problems.

4.41.10.1 Gravity Loads for an Axisymmetric Shell Problem

m (number of rings) and n (number of harmonics) are extracted from the /SYSTEM/ common block. The first m points in the BGPDT define the "zero" harmonic. The second m entries define the "one" harmonic etc. The acceleration vectors are calculated by the formulae:

$$g = \sqrt{g_x^2 + g_y^2 + g_z^2}\ , \tag{26}$$

$$g_{xy} = \sqrt{g_x^2 + g_y^2}\ , \tag{27}$$

$$\cos\theta_g = \frac{g_z}{g}\ , \tag{28}$$

$$\sin\theta_g = \frac{g_{xy}}{g}\ , \tag{29}$$

$$\sin\phi_g = \frac{g_y}{g_{xy}}\ , \tag{30}$$

$$\cos\phi_g = \frac{g_x}{g_{xy}}\ . \tag{31}$$

The vectors {a} for harmonics n = 0 and n = 1 are defined for load set cosine by:

$$\{a_0^c\} = g \cos\theta_g \begin{Bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \text{(all rings)}, \tag{32}$$

$$\{a_1^c\} = g \sin\theta_g \cos\phi_g \begin{Bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}; \tag{33}$$

(all rings)

and for load set sine by:

$$\{a_1^s\} = g \sin\theta_g \sin\phi_g \begin{Bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}. \tag{34}$$

These vectors are used as in the normal case. They are merged into $\{a_g\}$ which in turn is pre-multiplied by $[M_{gg}]$ to give the $\{P_g\}$ vector (subroutine GRAVL3).

4.41.10.2 Gravity Loads for Non-Shell Problems

The acceleration vector must be transferred to the global coordinate system at each grid point and expanded to a vector acting on the u_g coordinates. For each grid point (i) the BGPDT data is read, and using the CSTM data, a 3x3 basic to global transformation matrix $[T_i]$ is formed. The acceleration at the point i in basic coordinates is:

$$\{a_g^i\} = [T_i]\{g_b\}, \tag{35}$$

where $\{g_b\}$ is the gravity vector saved in Equation 25. The vector $\{a_g^i\}$ is placed in the total acceleration vector in positions SIL_i, SIL_i+1, and SIL_i+2. No values are calculated for scalar points or rotation coordinates (Subroutine GRAVL1).

When all $\{a_g\}$ vectors have been calculated for the whole structure, they are pre-multiplied by the structural mass matrix to produce a load vector:

$$\{P_g\} = [M_{gg}]\{a_g\}, \tag{36}$$

(subroutine SSG2B).

The gravity vectors are appended to the other load vectors, and scalar points are zeroed in case interaction occurred. Gravity loads on scalar points are not supported.

4.41.11 Subroutines

4.41.11.1 Subroutine Name: SSG1A.

1. Entry Point: SSG1A.

2. Purpose: To build a list of external loads, thermal loads and enforced deformation loads selected by the user in CASECC.

3. Calling Sequence: CALL SSG1A (N1,ILIST,NEDT,NTEMP,NCENT,CASECC,IHARM)

N1 - Number of external loads present - integer - output.

ILIST - List of load ID's with selected load ID's set negative - integer - output.

NEDT - Number of enforced deformation loads - integer - output.

NTEMP - Number of thermal loads - integer - output.

NCENT - Not used (set to zero) - integer - output.

CASECC - GINØ number of Case Control Data Block - integer - input.

IHARM - Boundary conditions for axisymmetric shell problem.

1 = sine, 2 = cosine - integer - output.

CØMMØN//XX,LØADNN

LØADNN - Number of records in CASECC to skip before beginning to build loads - integer - input.

CØMMØN/LØADX/

LØADX - See description of /LØADX/ common block below (section 4.41.11.8).

4.41.11.2 Subroutine Name: EXTERN.

1. Entry Point: EXTERN

2. Purpose: To compute user-selected external loads.

3. Calling Sequence: CALL EXTERN (NLIST,NGRAV,GVECT,ILIST,PG,N1,IHARM)

   NLIST - Number of load id's in ILIST array - integer - input.

   NGRAV - Number of gravity loads selected - integer - output.

   GVECT - Array of gravity vectors, 3 numbers per vector - real - output.

   ILIST - List of all load id's - integer - input.

   PG - Matrix control block for file on which external loads will be written - integer - input/output.

   N1 - Number of external load id's - integer - input.

   IHARM - Boundary condition for axisymmetric shell problem
   1 = sine, 2 = cosine.

   COMMON//LUSET

   LUSET - Length of PG - integer - input.

   COMMON/LOADX/ - See /LOADX/ description in section 4.41.11.8.

4.41.11.3 Subroutine Name: TEMPL.

1. Entry Point: TEMPL

2. Purpose: To compute thermal loads for each element.

3. Calling Sequence: CALL TEMPL (NTEMP,ILIST (N1+1),PG(1))

   NTEMP - Number of thermal loads - integer - input.

   ILIST(N1+1) - Beginning of thermal load list - integer - input.

   PG - Matrix control block for load file - integer - input/output.

   COMMON/LOADX/ - See /LOADX/ description in section 4.41.11.8.
   COMMON//LUSET

   LUSET - Length of PG - integer - input.

4.41.11.4 Subroutine Name: EDTL.

1. Entry Point: EDTL

2. Purpose: To compute enforced deformation loads for each element.

3. Calling Sequence: CALL EDTL (NEDT,ILIST(N1+1),PG(1))

NEDT - Number of enforced deformation loads - integer - input.

The remainder of the variables has the same meaning as in TEMPL (section 4.41.11.3).

4.41.11.5 Subroutine Name: GRAVL1.

1. Entry Point: GRAVL1

2. Purpose: To build acceleration vectors for gravity loads.

3. Calling Sequence: CALL GRAVL1 (NGRAV,GVECT,SCR1,IHARM)

NGRAV - Number of gravity loads selected - integer - input.

GVECT - Array of gravity vectors, three words per gravity vector - real - input.

SCR1 - GINØ file number on which to build the acceleration vectors - integer - input.

IHARM - Boundary condition for axisymmetric shell problem.
1 = sine, 2 = cosine - integer - output.

4.41.11.6 Subroutine Name: GRAVL2.

1. Entry Point: GRAVL2

2. Purpose: To add gravity loads onto previously generated load vectors and check scalar points.

3. Calling Sequence: CALL GRAVL2 (NGRAV,PGG,PG(1))

NGRAV - Number of gravity vectors - integer - input.

PGG - GINØ file number of gravity loads - integer - input.

PG - Matrix control block for all other (non-gravity) loads - integer - input/output.

CØMMØN/LØADX/

LØADX - See /LØADX/ common block (section 4.41.11.8).

CØMMØN//LUSET

LUSET - Length of PG vector - integer - input.

4.41.11.7 Subroutine Name: CØMBIN.

1. Entry Point: CØMBIN
2. Purpose: To combine subloads into subcase loads.
3. Calling Sequence: CALL CØMBIN (PG,ILIST,N1)

PG - Matrix control block for PG vector - integer - input/output.

ILIST - List of all load ID's - integer - input.

N1 - Number of entries in ILIST - integer - input.

CØMMØN/LØADX/

LØADX - See /LØADX/ description, section 4.41.11.8)

CØMMØN//LUSET

LUSET - Length of PG vector - integer - input.

CØMMØN/LØADS/NLØADS,IARY(300)

NLØADS - Number of loads to build.

IARY - For each load:

- Number of subloads
- Subload ID, Scale Factor } repeated for each subload
- Subload ID, Scale Factor } repeated for each subload

} repeated for each load

4.41.11.8 Common Block LØADX.

1. Purpose: To transmit file numbers and pointers.
2. Variable List: CØMMØN/LØADX/LC,SLT,BGPDT,ØLD,CSTM,SIL,ISIL,EST,MPT,GPT,EDT,IMPT,IGPTT, IEC,LØADF,MGG,NØBLD,DIT,ICM

SLT,BGPDT,CSTM,SIL,EST,MPT,GPTT,EDT,MGG,DIT - GINØ file numbers for respective data blocks - integer - input.

LC - Length of open core - integer - input.

ØLD - Current grid point position of the BGPDT - integer - input.

ISIL - Current SIL position - integer.

LØADF - GINØ file number of load file - integer - input.

NØBLD - Build-nobuild flag for direct load routines - integer.

IMPT - }
IGPTT - } Unused at present.
IEC - }

4.41.11.9 Subroutine Name: DIRECT.

1. Entry Point: DIRECT
2. Purpose: To apply loads due to FØRCE and MØMENT cards.
3. Calling Sequence: CALL DIRECT

   CØMMØN/LØADX/

   LØADX - See description of /LØADX/ above (section 4.41.11.8).

4.41.11.10 Subroutine Name TPØNT.

1. Entry Point: TPØNT
2. Purpose: To apply loads due to FØRCE1 and MØMENT1 cards.
3. Calling Sequence: CALL TPØNT

   CØMMØN/LØADX/

   LØADX - See description of /LØADX/ above (section 4.41.11.8).

4.41.11.11 Subroutine Name: FPØNT.

1. Entry Point: FPØNT

2. Purpose: To apply loads due to FØRCE2 and MØMENT2 cards.

3. Calling Sequence: CALL FPØNT

   CØMMØN/LØADX/

   LØADX - See description of /LØADX/ above (section 4.41.11.8).

4.41.11.12 Subroutine Name: SLØAD.

1. Entry Point: SLØAD

2. Purpose: To apply loads due to SLØAD cards.

3. Calling Sequence: CALL SLØAD

   CØMMØN/LØADX/

   LØADX - See description of /LØADX/ above (section 4.41.11.8).

4.41.11.13 Subroutine Name: PLØAD.

1. Entry Point: PLØAD

2. Purpose: To apply loads due to PLØAD cards.

3. Calling Sequence: CALL PLØAD

   CØMMØN/LØADX/

   LØADX - See description of /LØADX/ above (section 4.41.11.8).

4.41.11.14 Subroutine Name: RFØRCE.

1. Entry Point: RFØRCE

2. Purpose: To apply loads due to RFØRCE cards.

3. Calling Sequence: CALL RFØRCE (LCØRE)

   LCØRE - Current buffer top - integer - input.

   CØMMØN/LØADX/

   LØADX - See description of /LØADX/ above (section 4.41.11.8).

4.41.11.15 Subroutine Name: PRESAX.

1. Entry Point: PRESAX

2. Purpose: To apply loads due to axisymmetric pressure loads.

3. Calling Sequence: CALL PRESAX (IHARM)

IHARM - Axisymmetric boundary condition - integer - input.

CØMMØN/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.8).

4.41.11.16 Subroutine Name: GRAV.

1. Entry Point: GRAV

2. Purpose: To extract gravity vector and convert to basic coordinates.

3. Calling Sequence: CALL GRAV (NGRAV,GVECT,NEX,ILIST,NLØØP)

NGRAV - Number of gravity loads - integer - output.

GVECT - Array of gravity vectors - real - output.

NEX - Number of external loads - integer - input.

ILIST - List of external load ID's - integer - input/output.

NLØØP - Current pointer into ILIST.

CØMMØN/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.8).

4.41.11.17 Subroutine Name: PERMUT.

1. Entry Point: PERMUT

2. Purpose: To reorder a list of grid point ID's to allow the most efficient extraction of these grid points from the BGPDT.

3. Calling Sequence: CALL PERMUT (PØNT,IØRD,NP,ØLD)

PØNT - List of points - integer - input.

IØRD - Pointers to PØNT, i.e., IØRD (1) contains subscript of PØNT which should be extracted first from the BGPDT - integer - output.

NP - Number of points - integer - output.

ØLD - Current position of BGPDT - integer - input.

4.41.11.18 Subroutine Name: FNDPNT.

1. Entry Point: FNDPNT
2. Purpose: To extract BGPDT data from the BGPDT.
3. Calling Sequence: CALL FNDPNT (BGPDD,PØNT)

   BGPDD - Four-word BGPDT entry - output.

   PØNT - Grid point id of desired BGPDT entry - integer - input.

4.41.11.19 Subroutine Name: CRØSS.

1. Entry Point: CRØSS
2. Purpose: To compute the cross product of two vectors.
3. Calling Sequence: CALL CRØSS (V1,V2,V3)

   V1 - Three-word vector - real - input.

   V2 - Three-word vector - real - input.

   V3 = V1 x V2 - real - output.

4.41.11.20 Subroutine Name: NØRM.

1. Entry Point: NØRM
2. Purpose: To normalize a vector.
3. Calling Sequence: CALL NØRM (V1, XLV)

   V1 - Three-word vector - real - input/output.

   XLV - Norm of V1.

V1 on output = V1/XL unless XL = 0.0, then V1 = V1.

4.41.11.21 Subroutine Name: FNDSIL.

1. Entry Point: FNDSIL

2. Purpose: To find the SIL value of a particular grid point id.

3. Calling Sequence: CALL FNDSIL (GPID)

GPID - Grid point id on input, SIL value on output - integer - input/output.

COMMON/LOADX/

LOADX - See description of /LOADX/ above (section 4.41.11.8).

4.41.11.22 Subroutine Name: BASGLB.

1. Entry Point: BASGLB

2. Purpose: To convert a vector from the basic to the global coordinate system.

3. Calling Sequence: CALL BASGLB (VIN,VOUT,GRDPNT,CSYS)

VIN - Three-word input vector - real - input.

VOUT - Three-word output vector - real - output.

VIN may equal VOUT.

GRDPNT - Location of grid point at which vector is to be applied (not used unless coordinate system type is spherical or cylindrical) - real - input.

CSYS - Coordinate system id - integer - input.

COMMON/LOADX/

LOADX - See description of /LOADX/ above (section 4.41.11.8).

4.41.11.23 Subroutine Name: GLBBAS.

1. Entry Point: GLBBAS

2. Purpose: To convert a vector from global to basic coordinates.

3. Calling Sequence: CALL GLBBAS (VIN,VOUT,GRDPNT,CSYS)

Where the variables have the same names as in BASGLB.

4.41.11.24 Subroutine Name: MPYL.

1. Entry Point: MPYL

2. Purpose: To multiply two in-core matrices together: [A] · [B] = [C].

3. Calling Sequence: CALL MPYL (A,B,NCØLA,NRØWA,NCØLB,C)

   A (NCØLA,NRØWA) matrix
   B (NCØLB,NCØLA) matrix } These matrices are stored row-wise.
   C (NCØLB,NRØWA) matrix

Note: A or B ≠ C

4.41.11.25 Subroutine Name: MPYLT.

1. Entry Point: MPYLT

2. Purpose: To multiply two in-core matrices together $[A]^T$ · [B] = [C].

3. Calling Sequence: CALL MPYLT (A,B,NRØWB,NCØLA,NCØLB,C)

   A (NCØLA,NRØWB) matrix
   B (NCØLB,NRØWB) matrix } These matrices are stored row-wise.
   C (NCØLB,NCØLA) matrix

4.41.11.26 Subroutine Name: FDCSTM.

1. Entry Point: FDCSTM

2. Purpose: To extract the orientation matrix from the CSTM.

3. Calling Sequence: CALL FDCSTM (ICSTM)

   ICSTM - Coordinate system id desired.

   CØMMØN/TRANX/XX(5),TØ(3,3)

   TØ - 3x3 orientation matrix.

   CØMMØN/LØADX/

   LØADX - See description of /LØADX/ above (section 4.41.11.8).

4.41.11.27 Subroutine Name: FGPTT.

1. Entry Point: FGPTT

2. Purpose: To find temperature in the GPTT.

3. Calling Sequence: CALL FGPTT (SILAR,TAR,NP)

   SILAR - Array of SIL's for which temperatures are desired - integer - input.

   TAR - Array of temperatures at SIL's - real - output.

   NP - Number of entries in SILAR.

   CØMMØN/FPT/TØ,NSIL,NGPTT,NRØWSP

   TØ - Default temperature - real - input.

   NSIL - Number of SIL entries - integer - input.

   NGPTT - Length of GPTT.

   NROWSP - Beginning of SIL table - integer - input.

4.41.11.28 Subroutine Name: FEDT.

1. Entry Point: FEDT

2. Purpose: To extract one enforced deformation from the EDT given one element id.

3. Calling Sequence: CALL FEDT (EID,DELTA,IDEFM)

   EID - Element id - integer - input.

   DELTA - Deformation of element EID - real - output.

   IDEFM - Set id of current deformation set.

4.41.11.29 Subroutine Name: GRAVL3.

1. Entry Point: GRAVL3

2. Purpose: To compute an acceleration vector for an axisymmetric shell problem.

3. Calling Sequence: CALL GRAVL3 (NGRAV,GVECT,AG,IHARM)

   NGRAV - Number of gravity vectors - integer - input.

   GVECT - Gravity vector array - real - input.

   AG - GINØ file number of acceleration vector - integer - input.

IHARM - Boundary condition flag

1 = sine, 2 = cosine

CØMMØN//LUSET

LUSET - Length of PG vector.

CØMMØN/SYSTEM/IX(26),MN

MN - Packed word giving number of rings/number of harmonics.

4.41.11.30 Subroutine Name: TTRIRG

1. Entry Point: TTRIRG

2. Purpose: To calculate an element thermal load vector for a triangular cross-section ring in the SSG1 module.

3. Calling Sequence: CALL TTRIRG (TI,PG)

TI - Array of four temperatures at the four points of the ring - real - input.

PG - Load vector array - real - input.

CØMMØN/TRIMEX/

TRIMEX - This contains the EST entry for the element.

4.41.11.31 Subroutine Name: TTRAPR.

1. Entry Point: TTRAPR

2. Purpose: To calculate an element thermal load vector for a trapezoidal cross-section ring in the SSG1 module.

3. Calling Sequence: CALL TTRAPR (TI, PG)

CØMMØN/TRIMEX/

The arguments are the same as in TTRIRG.

4.41.11.32 Subroutine Name: TTØRDR.

1. Entry Point: TTØRDR

2. Purpose: To calculate an element thermal load vector for a toroidal thin shell ring in the SSG1 module.

3. Calling Sequence: CALL TTØRDR (TI,PG)

CØMMØN/TRIMEX/

The arguments are the same as those in TTRIRG (section 4.41.11.30).

4.41.11.33 Subroutine Name: FCURL.

1. Entry Point: FCURL

2. Purpose: To form the element thermal load matrices in field coordinates for the toroidal thin shell ring in subroutine TTØRDR.

3. Calling Sequence: CALL FCURL (FME0,FME1,FFE0,FFE1,YI,S,LAM1)

FME0, FME1
FFE0, FFE1 — The resultant thermal load matrices.

YI - Array of integral values.

S, LAM1 - Terms used in the evaluation of the thermal load matrices.

4.41.11.34 Subroutine Name: CØNE.

1. Entry Point: CØNE

2. Purpose: To calculate an element thermal load vector for an axisymmetric shell in the SSG1 module.

3. Calling Sequence: CALL CØNE (TI,PG)

CØMMØN/TRIMEX/

The arguments are the same as in TTRIG (section 4.41.11.30).

4.41.11.35 Subroutine Name: QDMEM.

1. Entry Point: QDMEM

2. Purpose: To calculate an element thermal load vector for quadrilateral elements in the SSG1 module.

3. Calling Sequence: CALL QDMEM (TI,PG)

CØMMØN/TRIMEX/

The arguments are the same as in TTRIRG (section 4.41.11.30).

4.41.11.36 Subroutine Name: TRIMEM.

1. Entry Point: TRIMEM

2. Purpose: To calculate an element thermal load vector for triangular elements in the SSG1 module.

3. Calling Sequence: CALL TRIMEM (TYPE,TBAR,PG)

CØMMØN/TRIMEX/

TYPE - Flag indicating normal entry (0) or subelement call from a QDMEM (1) - integer - input.

TBAR - Average temperature over the element - real - input.

PG and /TRIMEX/ are as in TTRIRG (section 4.41.11.30).

4.41.11.37 Subroutine Name: BAR.

1. Entry Point: BAR

2. Purpose: To calculate an element thermal load vector or deformation load vector for the bar element in the SSG1 module.

3. Calling Sequence: CALL BAR (PG, IDEFM, ITEMP, IDEFT)

CØMMØN/TRIMEX/

IDEFM - 0 if element deformation load vector is not to be computed - integer - input.

ITEMP - 0 if element thermal load vector is not to be computed - integer - input.

IDEFT - Set ID of the element deformation set. This is used only if IDEFM ≠ 0 - integer - input.

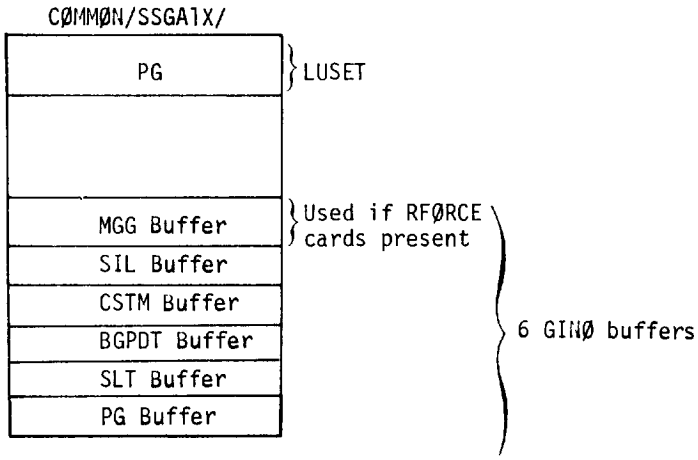
PG,/TRIMEX/ are as in section 4.41.11.30.

4.41.12 Design Requirements

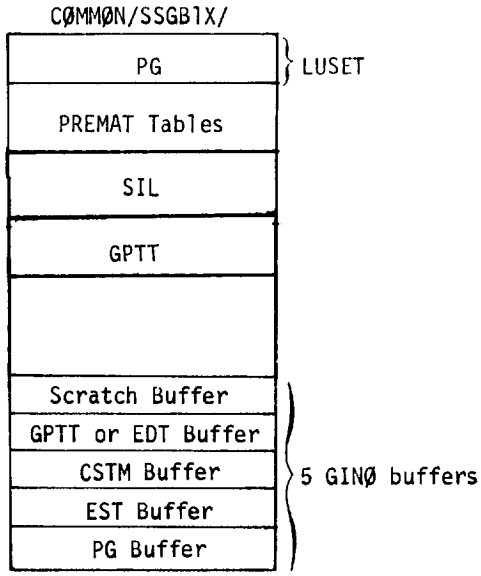
Three scratch files are needed.

Open core is defined as follows:

1. /SSGA1X/ during EXTERN phase



2. /SSGB1X/ during EDTL, TEMPL phase



100 LØAD ID's maximum.

4.41.13 Diagnostic Messages

If the core storage requirements as depicted in the above diagrams are not met, SSG1 will issue fatal error message 3008.

4.42 FUNCTIONAL MODULE SSG2 (STATIC SOLUTION GENERATOR - PHASE 2)

4.42.1 Entry Point: SSG2

4.42.2 Purpose

To reduce the applied load vectors and enforced displacements into equivalent load vectors applied to the independent displacement coordinate sets.

4.42.3 DMAP Calling Sequence

```
SSG2  USET,GM,YS,KFS,GØ,DM,PG/QR,PØ,PS,PL/  $
```

4.42.4 Input Data Blocks

USET - Displacement set definitions table.

GM - Multipoint constraint transformation matrix - m set.

YS - Constrained displacements - s set.

KFS - Partition of stiffness matrix after single-point constraints have been removed.

GØ - Structural matrix partitioning transformation matrix.

DM - Rigid body transformation matrix.

PG - Static load vector matrix giving static loads - g set.

Notes:
1. USET must be present.
2. GM must be present if m set is not null.
3. YS must be present if s set is not null.
4. KFS must be present if s set is not null.
5. GØ must be present if o set is not null.
6. DM must be present if r set is not null.
7. PG must be present.

4.42.5 Output Data Blocks

QR - Determinate support forces matrix - r set.

PØ - Partition of the load vector matrix giving loads due to static force - o set.

PS - Partition of load vector matrix giving loads in s set.

PL - Partition of load vector matrix giving static loads on ℓ set.

Notes: 1. QR must be present if r set is non-null.

2. PØ must be present if o set is non-null.

3. PS must be present if s set is non-null.

4. PL must be present if ℓ set is non-null.

5. If the problem has no sets, SSG2 will return.

4.42.6 Parameters

None

4.42.7 Method

The fifth word of the USET trailer control block is analyzed to determine the presence of m's, s's, o's, or r's.

Each of the following steps is omitted if the appropriate set is null. The following steps indicate the operations on one load set vector. The actual algorithm uses all vectors in each step by performing matrix operations.

1. If m's are present, the PG vectors are partitioned using USET (UG,UN,UM) and subroutines CALCV and SSG2A:

$$\{P_g\} \Longrightarrow \left\{ \begin{array}{c} \overline{P}_n \\ \hline P_m \end{array} \right\} \quad . \tag{1}$$

The loads on the u_n set are calculated as:

$$\{P_n\} = \{\overline{P}_n\} + [G_m]^T \{P_m\} \quad , \tag{2}$$

by calling subroutine SSG2B.

2. If s's are present, the $\{P_n\}$ load vectors are partitioned using USET (UN,UF,US) and subroutines CALCV, and SSG2A:

$$\{P_n\} \Longrightarrow \left\{ \begin{array}{c} \overline{P}_f \\ \hline P_s \end{array} \right\} \tag{3}$$

The $\{P_s\}$ vectors are output in data block PS. The loads on the u_f set are calculated as:

$$\{P_f\} = \{\bar{P}_f\} - [K_{fs}] \{Y_s\} \quad . \tag{4}$$

The $\{Y_s\}$ vector normally is zero. If more than one load vector is being reduced, $\{Y_s\}$ is expanded to have N identical columns such that the above matrix equation is dimensionally consistent. This is accomplished in subroutine SSG2B1 (see below).

3. If o's are present, the $\{P_f\}$ vectors are partitioned using USET (UF,UA,UØ) and subroutines CALCV and SSG2A:

$$\{P_f\} \Rightarrow \begin{Bmatrix} \bar{P}_a \\ \hline P_o \end{Bmatrix} \quad . \tag{5}$$

$\{P_o\}$ vectors are written on data block PØ.

The equivalent loads on the u_a set are calculated as:

$$\{P_a\} = \{\bar{P}_a\} + [G_o]^T \{P_o\} \quad . \tag{6}$$

Subroutine SSG2B is called to perform this operation.

4. If r's are present, the $\{P_a\}$ vectors are partitioned using USET (UA,UL,UR) and subroutines CALCV and SSG2A:

$$\{P_a\} \Rightarrow \begin{Bmatrix} P_\ell \\ \hline P_r \end{Bmatrix} . \tag{7}$$

$\{P_\ell\}$ vectors are written on data block PL.

The reaction vectors on the support points are:

$$\{q_r\} = - \{P_r\} - [D_m]^T \{P_f\} \tag{8}$$

where $[D_m]$ corresponds to the data block DM. $\{q_r\}$ vectors are written on data block QR.

4.42.8 Subroutines

SSG2 uses matrix subroutines CALCV and SSG2A for matrix partitioning operations and subroutine SSG2B to drive subroutine MPYAD. See section 3 for details.

4.42.8.1 Subroutine Name: SSG2B1

1. Entry Point: SSG2B1

2. Purpose: SSG2B1 is exactly like SSG2B. It computes [A] [B] + [C] = [D] except YS is first expanded onto SCR1 which then plays the role of [B] .

3. Calling Sequence: CALL SSG2B1 (A,SCR1,C,D,T,PREC,ISIGN,YS,SCR2)

The variables have the meaning as SSG2B except for YS, which is the GINØ file number of the YS data block - integer - input. See 3.5.13 for SSG2B details.

4.42.9 Design Requirements

/SSG2X/ is open core for CALCV. Four scratch files are used.

SSG2B1 is an entry point in SSG2B.

4.43 FUNCTIONAL MODULE SSG3 (STATIC SOLUTION GENERATOR - PHASE 3)

4.43.1 Entry Point: SSG3

4.43.2 Purpose:

To perform the actual static solutions. A displacement solution is produced for each applied load and tested for possible matrix decomposition errors.

4.43.3 DMAP Calling Sequence

```
SSG3  LLL,ULL,KLL,PL,LØØ,UØØ,KØØB,PØ/ULV,UØØV,RULV,RUØV/V,N,ØMIT/V,Y,IRES    $
```

4.43.4 Input Data Blocks

LLL - Lower triangular factor of KLL - ℓ set.

ULL - Upper triangular factor of KLL - ℓ set.

KLL - Partition of stiffness matrix - ℓ set.

PL - Partition of the load vector matrix giving static loads on ℓ set

LØØ - Lower triangular factor of KØØB - o set.

UØØ - Upper triangular factor of KØØB - o set.

KØØB - Partition of stiffness matrix - o set.

PØ - Partition of the load vector matrix giving loads due to static forces - o set.

Notes:
1. ULL,LLL and PL must be present.
2. KLL can be purged if RULV is purged.
3. UØØ, LØØ, PØ can be purged if ØMIT < 0.
4. KØØB can be purged if ØMIT < 0 or RUØV is purged.

4.43.5 Output Data Blocks

ULV - Partition of the displacement vector matrix giving displacements - ℓ set.

UØØV - Partition of the displacement vector matrix giving displacements - o set.

RULV - Residual vector matrix for the ℓ set.

RUØV - Residual vector matrix for the o set.

Notes:
1. ULV must be present.

2. UØØV can be purged if ØMIT < 0.
3. RULV and RUØV can be purged.
4. [RULV] = [KLL] [ULV] - [PL].
   [RUØV] = [KØØB][UØØV] - [PØ].

4.43.6 Parameters

ØMIT - Input-integer-no default. ØMIT controls operations on o-set matrices.

IRES - Not used-integer-no default. IRES is a user-controlled parameter that he may set to +1 on a Bulk Data PARAM card so that residual vectors may be printed. It is included here to define an initial value for IRES.

4.43.7 Method

4.43.7.1 Solution Algorithim

For normal statics problems PL and PØ vectors are used; for inertia relief problems the PLI and PØI vectors are generated in the SSG4 module and used instead of PL and PØ. The equations to be solved are:

$$[K_{\ell\ell}]\ \{u_\ell\} = \{P_\ell\}, \tag{1}$$

and if o's are present (ØMIT $\geq$ 0)

$$[K_{oo}]\ [u_o^o] = \{P_o\}, \tag{2}$$

where

$$[K_{\ell\ell}] = [L_{\ell\ell}]\ [U_{\ell\ell}]\ , \tag{3}$$

$$[K_{oo}] = [L_{oo}]\ [U_{oo}]\ . \tag{4}$$

$[L_{\ell\ell}]$ and $[L_{oo}]$ are lower traingular matrices. $[U_{\ell\ell}]$ and $[U_{oo}]$ are upper triangular matrices. The equations are solved in the following manner:

$$[L]\ [U]\ \{u\} = \{P\}, \tag{5}$$

or

$$[L]\ \{y\} = \{P\}\ . \tag{6}$$

In turn the solution for the displacement vector is:

$$[U]\ \{u\} = \{y\}\ . \tag{7}$$

4.43.7.2 Error Check Algorithm

If RULV is not purged, an error check is made. Since it is possible for the stiffness matrices to be nearly singular or ill-conditioned, the module calculates the following error analysis terms for both the $\{u_\ell\}$ and $\{u_o\}$ solutions:

$$\{\delta P_\ell\} = \{P_\ell\} - [K_{\ell\ell}]\ \{u_\ell\}\ , \tag{8}$$

$$\varepsilon_\ell = \frac{\{u_\ell\}^T\ \{\delta P_\ell\}}{\{P_\ell\}^T\ \{u_\ell\}}\ . \tag{9}$$

$\{\delta P_\ell\}$ forms data block RULV or RUØV. ε_ℓ is printed for each solution.

4.43.8 Subroutines

SSG3A - is the only auxiliary subroutine in module SSG3 (see section 3.5.18 for description).

4.43.9 Design Requirements

Two scratch files are needed.

4.44 FUNCTIONAL MODULE SSG4 (STATIC SOLUTION GENERATOR - PHASE 4).

4.44.1 Entry Point: SSG4

4.44.2 Purpose

The purpose of this module is to calculate mass loads in a Static Analysis with Inertia Relief problem. The rigid body accelerations are functions of the reactions on the fictitious supports. The inertia loads on the structure are proportional to these accelerations.

4.44.3 DMAP Calling Sequence

```
SSG4    PL,QR,PØ,MR,MLR,DM,MLL,MØØB,MØAB,GØ,USET/PLI,PØI/V,N,ØMIT  $
```

4.44.4 Input Data Blocks

PL - Partition of the load vector matrix giving static loads - ℓ set.

QR - Determinate support forces matrix - r set.

PØ - Partition of the load vector matrix giving loads due to static force - o set.

MR - Rigid body mass matrix - r set.

MLR - Partition of mass matrix.

DM - Rigid body transformation matrix.

MLL - Partition of mass matrix - ℓ set.

MØØB - Partition of mass matrix - o set.

MØAB - Partition of mass matrix.

GØ - Structural matrix partitioning transformation matrix.

USET - Displacement set definitions table.

Note: All matrices must be present if their appropriate set is non-null.

4.44.5 Output Data Blocks

PLI - Partition of load vector for inertia relief matrix giving loads due to static and inertial forces on ℓ set.

PØI - Partition of load vector for inertia relief matrix giving loads due to inertial and static forces on o set.

Note: Both matrices must be present if their appropriate set is non-null.

4.44.6 Parameters

ØMIT - Input-integer-no default. If ØMIT > 0, the o set is non-null.

4.44.7 Method

1. The accelerations of the u_r degrees of freedom are:

$$\{a_r\} = - [m_r]^{-1} \{q_r\} . \tag{1}$$

$[m_r]$ corresponds to data block MR. Subroutines FACTØR and SSG3A are used to solve for $\{a_r\}$.

2. The total load vectors on the structure are:

$$\{P_\ell^i\} = \{P_\ell\} + \left[[M_{\ell\ell}] [D] + [M_{\ell r}]\right] \{a_r\}. \tag{2}$$

[D] corresponds to data block DM. Subroutine SSG2B is used to drive MPYAD.

3. If ØMIT ≥ 0:

$$\{P_o^i\} = \{P_o\} + \left[[M_{oo}] [G_o] + [M_{oa}]\right] \left[\frac{D}{I}\right] \{a_r\}. \tag{3}$$

The product $\left[\frac{D}{I}\right] \{a_r\}$ is formed by merging columns of [D] $\{a_r\}$ with $\{a_r\}$ using USET (UA,UL,UR). Subroutine SDR1B is used to drive MERGE, and SSG2B for the matrix products.

4.44.8 Subroutines

SSG2B - See subroutine description, section 3.5.13

SSG3A - See subroutine description, section 3.5.18

SDR1B - See subroutine description, section 3.5.8

4.44.9 Design Requirements

Five scratch files are necessary.

4.45 FUNCTIONAL MODULE SDR1 (STRESS DATA RECOVERY - PHASE 1)

4.45.1 Entry Point: SDR1

4.45.2 Purpose

The SDR1 module utilizes solution vectors to produce displacements, eigenvectors, velocities, accelerations, applied loads and reaction loads. The vectors input to SDR1 are in the form of packed matrices with each column a solution vector for a different subcase, eigenvalue, frequency/load, or transient output time. The row position of each term in a vector corresponds to a degree of freedom in a unique displacement set. The relative position of the term must be converted to a relative position in the vector which includes all displacement components in the system. The dependent components of the displacement vector are recovered and merged to produce a complete vector describing all degrees of freedom in the structural or dynamics model. In the Static Analysis or Static Analysis with Inertia Relief Rigid Formats, SDR1 collects solutions for each boundary condition onto a single file, convenient for the solution of symmetry problems.

4.45.3 DMAP Calling Sequence

```
SDR1  USET,PG,ULV,UØØV,YS,GØ,GM,PS,KFS,KSS,QR/UGV,PGG,QG/V,N,APPEND/V,N,FØRMAT  $
```

4.45.4 Input Data Blocks

USET - Displacement set definitions table.

PG - Static load vector matrix giving static loads - g set.

ULV - Partition of the displacement vector matrix giving displacements - ℓ set.

UØØV - Partition of the displacement vector matrix giving displacements in the o set.

YS - Constrained displacements - s set.

GØ - Structural matrix partitioning transformation matrix.

GM - Rigid body transformation matrix.

PS - Partition of load vector matrix giving loads in s set.

KFS - Partition of stiffness matrix after single-point constraints have been removed - s set.

KSS - Partition of stiffness matrix after single-point constraints have been removed - s set.

QR - Determinate support forces matrix - r set.

Notes: 1. USET must always be present.

2. PG may or may not be present.

3. ULV must always be present.

4. UØØV must be present unless the o set is null or FØRMAT = DYNAMICS (see below).

5. YS may or may not be present.

6. GØ must be present unless the o set is null.

7. GM must be present unless the m set is null.

8. PS may or may not be present.

9. KFS must be present unless the s set is null or QG is not present.

10. KSS must be present unless YS is absent or the s set is null or QG is not present.

11. QR may or may not be present.

4.45.5 Output Data Blocks

UGV - Displacement vector matrix giving displacements in the g set.

PGG - Static load vector appended to include all boundary conditions - g set.

QG - Single-point constraint forces and determinate support forces matrix - g set.

Notes: 1. If PG is present, PGG must be present.

2. UGV must be present.

3. QG must be present.

4. If APPEND $\neq$ 1, UGV, PGG, QG will be appended to the data already on these files.

4.45.6 Parameters

APPEND - Input-integer- no default. See note 4 above.

FØRMAT - Input-BCD-no default. Format indicates the problem type.

STATICS - Statics type problem.

REIGEN - Real eigenvalue problem.

DYNAMICS - Dynamic problem.

4.45.7 Method

The following steps are performed by the SDR1 module in the most general case. Most problems, however, do not use all of the constraint options. In these cases certain steps

are skipped. The task SDR1 performs also varies from Rigid Format to Rigid Format.

1. If PG is present, it is copied or appended onto PGG. (Subroutine SDR1A).

2. If r's are present and the problem type is not DYNAMICS or REIGEN:

$$\left\{\frac{u_\ell}{o}\right\} \Rightarrow \left\{u_a\right\} . \tag{1}$$

This is accomplished by subroutine SDR1B, the driver for MERGE. See section 3.5.8 for details.

If s's are also present:

$$\left\{\frac{q_r}{o}\right\} \Longrightarrow \left\{q_f\right\} . \tag{2}$$

Otherwise,

$$\left\{\frac{q_r}{o}\right\} \Longrightarrow \left\{q_g\right\} , \tag{3}$$

which is then copied or appended onto QG. (Subroutine SDR1B and SDR1A).

3. If o's are present, the dependent degrees of freedom of the "omitted" coordinates $\{u_o\}$ are computed. For statics problems:

$$\left\{u_o\right\} = \left[G_o\right] \left\{u_a\right\} + \left\{u_o^o\right\} . \tag{4}$$

For Dynamic problems:

$$\left\{u_o\right\} = \left[G_o^d\right] \left[u_d\right] . \tag{5}$$

This is accomplished by calling SSG2B (see section 3.5.13). The independent degrees of freedom $\{u_a\}$ or $\{u_d\}$ are merged with the dependent coordinates $\{u_o\}$, using subroutine SDR1B:

$$\left\{\frac{u_a}{u_o}\right\} \Longrightarrow \left\{u_f\right\} . \tag{6}$$

4. If s's are present, the coordinates fixed by single-point constraints, $\{u_s\}$,

may have constrained displacements, $\{Y_s\}$, in a statics problem. These are merged with the $\{u_f\}$ vector using subroutine SDR1B:

$$\left\{\begin{array}{c} u_f \\ \hline Y_s \end{array}\right\} \Longrightarrow \{u_n\} \tag{7}$$

The forces of single-point constraint $\{q_s\}$ are calculated using SSG2B from the equation:

$$\{q_s\} = -\{P_s\} + [K_{fs}]^T \{u_f\} + [K_{ss}] \{Y_s\}. \tag{8}$$

Subroutine SDR1B is used to merge $\{q_s\}$ with $\{q_f\}$ to form $\{q_g\}$. $\{q_f\}$ is $\{q_r\}$ expanded to the f-set:

$$\left\{\begin{array}{c} q_f \\ \hline q_s \end{array}\right\} \Longrightarrow \{q_g\} \cdot \tag{9}$$

$\{q_g\}$ is copied or appended onto data block QG (Subroutine SDR1A). If YS is not present,

$$\left\{\begin{array}{c} u_f \\ \hline o \end{array}\right\} \Longrightarrow \{u_n\} . \tag{10}$$

The forces of single-point constraint are computed as in Equation 7 except the $\{Y_s\}$ term is omitted.

If e-points (extra points) are present in a dynamics problem, $[K_{fs}]$ must be converted to $[K^d_{fs}]$ by subroutine SDR1C. If the problem is a transient problem, $\{u_f\}$ must be reduced to displacement vectors only, since $\{u_f\}$ contains triples - displacement, velocity and acceleration in transient problems. This reduction is accomplished in SDR1D.

5. If m's are present, the dependent coordinates, $\{u_m\}$, of the multipoint constraint equations are calculated using:

$$\{u_m\} = [G_m] \{u_n\} . \tag{11}$$

In dynamics problems $[G_m] = [G^d_m]$. The two vectors $\{u_m\}$ and $\{u_n\}$ are merged

$$\left\{\begin{matrix} u_n \\ \hline u_m \end{matrix}\right\} \Rightarrow \{u_g\} \ . \qquad (12)$$

6. $\{u_g\}$ (or $\{u_p\}$ for dynamics problems) is copied or appended onto UGV.

4.45.8 Subroutines

SSG2B and SDR1B are used as utility routines. See section 3.5.13 and 3.5.8 for details.

4.45.8.1 Subroutine Name: SDR1A

1. Entry Point: SDR1A

2. Purpose: To copy or append vectors.

3. Calling Sequence: CALL SDR1A (IN,IØUT)

IN - GINØ file number of input file - integer - input.

IØUT - GINØ file number of output file - integer - input.

CØMMØN//IAPEND- Append flag. If IAPEND ≠ 1, IØUT will be positioned after the last record and IN copied onto IØUT at this point. Otherwise IN is copied onto IØUT.

4.45.8.2 Subroutine Name: SDR1C

1. Entry Point: SDR1C

2. Purpose: To expand a file (KFS) from the f set to the fe set for dynamics.

3. Calling Sequence: CALL SDR1C (IPVECT,KFS,KDFS)

IPVECT - GINØ file number of the partitioning vector previously generated - integer - input.

KFS - GINØ file number of $[K_{fs}]$ - integer - input.

KDFS - GINØ file number of $[K^d_{fs}]$ - integer - input.

4. Design Requirement: SDR1C depends on a previous call to SDR1B to initialize set parameters.

4.45.8.3 Subroutine Name: SDR1D

1. Entry Point: SDR1D

2. Purpose: To strip velocity and acceleration vectors from $\{u_f\}$.

3. Calling Sequence: CALL SDR1D (PS,IUF,IUF1,ITRAN)

   PS - GINØ file number of PS - integer - input.

   IUF - GINØ file number of $\{u_f\}$ - integer - input.

   IUF1 - GINØ file number of $\{u_f\}$ stripped - integer - input.

   ITRAN - Flag which shows whether problem is a transient analysis or not - integer - input. Its values are:

   1 $\Longrightarrow$ not transient

   0 $\Longrightarrow$ transient.

4.45.9 Design Requirements

Six scratch files are required.

4.46 FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

4.46.1 Entry Point: SDR2

4.46.2 Purpose

The SDR2 module processes the output requests for forces of single-point constraint, loads, point displacements, point velocities, point accelerations, element stresses, and element forces, formatting the output data blocks with these final output results for: a) direct outputing by the Output File Processor (ØFP) module or b) input to the SØRT2 processor (SDR3) module and then the XY-output modules (XYTRAN and XYPLØT).

4.46.3 DMAP Calling Sequence

```
SDR2    {CASECC},CSTM,MPT,DIT,{EQEXIN},{SIL },GPTT,EDT,
        {CASEXX}              {EQDYN }{SILD}

BGPDT,{PGG  },{QG  },{UGV  },{EST },XYCDB /
      {PGV1 }{QBG }{UGV1 }{ESTL}
      {PPF  }{BQG }{UPV  }
      {PPT  }{QG1 }{UPVC }
      {LAMA }{QPC }{UBGV }
      {CLAMA}{QP  }{PHIG }
                   {CPHIP}

{ØPG1 },{ØQG1 },{ØUGV1 },{ØES1 },{ØEF1 },{PUPV  }
{ØPPC1}{ØQBG1}{ØUPV1 }{ØESC1}{ØEFC1}{PUGV1 }
{ØPP1 }{ØBQG1}{ØUPVC1}{ØESB1}{ØEFB1}{PPHIG }
       {ØQPC1}{ØUBGV1}{ØBES1}{ØBEF1}{PUBGV1} /
       {ØQP1 }{ØPHIG }
              {ØCPHIP}

C,N,{STATICS}
    {REIGEN }
    {DSO    }
    {DS1    }
    {FREQ   }
    {TRANSNT} /V,N,NØSØRT2  $
    {BLKO   }
    {BLK1   }
    {CEIGEN }
    {PLA    }
```

4.46.4 Input Data Blocks

CASECC - Case Control Data Table.

CASEXX - Case Control Data Table for Dynamics problems.

CSTM - Coordinate System Transformation Matrices.

MPT - Material Property Table.

DIT - Direct Input Tables.

EQEXIN - Equivalence between external grid or scalar numbers and internal numbers.

EQDYN - Equivalence between external points and scalar index values.

SIL - Scalar Index List.

SILD - Scalar Index List for Dynamics.

GPTT - Grid Point Temperature Table.

EDT - Element Deformation Table.

BGPDT - Basic Grid Point Definition Table.

PGG - Static load vector appended to include all boundary conditions.

PGV1 - Matrix of successive sums of incremental load vectors.

PPF - Dynamic loads for frequency response.

PPT - Linear dynamic loads for transient analysis.

LAMA - Real Eigenvalue Table.

CLAMA - Complex Eigenvalue Table.

QG - Single-point constraint forces and determinant support forces matrix.

QBG - Single-point forces of constraint matrix for Differential Stiffness - g set.

BQG - Single-point forces of constraint matrix for a Buckling Analysis problem - g set.

QG1 - Matrix of successive sums of incremental vectors of single-point constraint forces.

QPC - Complex single-point forces of constraint - p set.

QP - Transient single-point forces of constraint - p set.

UGV - Displacement vector matrix giving displacements in the g set.

UGV1 - Matrix of successive sums of incremental displacement vectors.

UPV - Transient solution vectors - p set.

UPVC - Frequency response solution vectors - p set.

UBGV - Displacement vector matrix for differential stiffness giving displacements in the g set.

PHIG - Eigenvector matrix giving eigenvectors.

CPHIP - Complex eigenvectors in the p set.

EST - Element Summary Table.

ESTL - Element Summary Table for Linear Elements.

XYCDB - XY Case Control Data Block.

Notes:

1. If the first input data block is purged, it is a fatal error. This data block is called "Case Control" in this Module Functional Description.

2. The CSTM may be purged if no coordinate systems are referenced, or if stresses and/or forces are not requested.

3. The MPT may be purged if no stress or force requests are present.

4. The DIT may be purged if no stress or force requests are present, or if no temperature dependent materials are referenced.

5. The second record of EQEXIN or EQDYN must exist if a request exists for any of: loads, forces of single-point constraint, displacements, velocities, accelerations, or plots.

6. SIL or SILD may be purged if no stress or force requests exist, or there are no extra-points and no thermal loads. (The second record is used by SDR2).

7. The GPTT may be purged if no thermal loading exists, or there are no requests for stresses or forces.

8. The EDT may be purged if there are no element requests for forces or stresses, or if there are no enforced element deformations in the problem.

9. The BGPDT may be purged if the problem is in basic coordinates and no element requests for stresses or forces exist.

10. LAMA or CLAMA may not be purged if an eigenvalue or frequency response problem.

11. If input data block 11 (QG or QBG etc.) is purged, forces of single-point constraint requests are ignored.

12. If input data block 12 (UGV or UGV1 etc.) is purged, SDR2 will process only loads and forces of single-point constraint requests.

13. If the EST or ESTL is purged, element stresses and force requests are ignored.

14. The XYCDB may be purged.

4.46.5 Output Data Blocks

ØPG1, ØPPC1, ØPP1 - Output load vector requests.

ØQG1, ØQBG1, ØBQG1, ØQPC1, ØQP1 - Output forces of single-point constraint requests.

ØUGV1, ØUPVC1, ØUBGV1 - Output displacement vector requests.

ØPHIG, ØCPHIP - Output eigenvector requests.

ØES1, ØESC1, ØESB1, ØBES1 - Output element stress requests.

ØEF1, ØEFC1, ØEFB1, ØBEF1 - Output element force requests.

PUPV, PUGV1, PPHIG, PUBGV1 - Translation components of the displacement vector rotated to basic coordinates.

Notes:

Output data blocks purged will result in output requests to those data blocks not being processed.

4.46.6 Parameters

STATICS - BCD constant indicating a Statics solution.

REIGEN - BCD constant indicating a Real Eigenvalue solution.

DSO - BCD constant indicating the Statics phase of a Differential Stiffness solution.

DS1 - BCD constant indicating final phase of a Differential Stiffness solution.

FREQ - BCD constant indicating a Frequency Response solution.

TRANSNT - BCD constant indicating a Transient Response solution.

BKLO - BCD constant indicating the Statics phase of a Buckling solution.

BKL1 - BCD constant indicating the final phase of a Buckling solution.

CEIGEN - BCD constant indicating a Complex Eigenvalue solution.

PLA - BCD constant indicating a Piecewise Linear Analysis solution.

NØSØRT2 - Integer-Output-Set to 0 if there are no SØRT2 requests or requirements, and set to 1 otherwise.

4.46.7 Method

The SDR2 functional module is constructed in a modular form consisting of five stages. A small executive control program (subroutine SDR2) containing the main entry point for the module serially calls the five stages for execution.

1. Stage I, performed by subroutine SDR2AA, prepares, if necessary, a modified Case Control data block, internal to SDR2, to insure that any XY-output requests present and not included in the ØFP output requests of Case Control are included in the output of SDR2 for later processing by functional modules XYTRAN and XYPLØT.

2. Stage II, performed by subroutine SDR2A, analyzes the overall output requests within the subcases of Case Control and sets flags for use by stages III, IV, and V.

3. Stage III, performed by subroutine SDR2B, is executed only if stage II has determined that some element force or stress output requests are present within Case Control. If executed, element stress matrices are computed once and stored along with certain other element properties appearing in the EST for each element appearing in a master set of element requests. (The master set is a set which is the union of all elements for which output requests are present in any subcase of Case Control). These stored data are used in stage V repeatedly as necessary to satisfy the various output request combinations and multiple displacement vectors that may be present.

4. Stage IV, performed by subroutine SDR2C, is executed only if in stage II it has been determined that some output requests are present for any of: forces of single-point constraint, loads, displacement, velocities, accelerations, or deformed structure plots. If they are present, these requests (except for structure plots) are fully processed within this stage.

5. Stage V, performed by subroutine SDR2D, is executed only if stage III was executed. Stage V performs final element stress and force computations. For each subcase of Case Control containing element output requests, the appropriate displacement vector is applied to the stress matrices, computed in Stage III, of the elements requested for output to arrive at the final stress and/or force outputs.

For several of the five stages the main subroutine listed utilizes additional subroutines to accomplish its particular task. The methods employed within each stage are further described in the subroutine descriptions in the next section.

4.46.8 Subroutines

The utility routines PRETRS and PREMAT are called within the SDR2 module for initialization purposes so that the structural element subroutines can call the entry point TRANSS of PRETRS and MAT of PREMAT to fetch Coordinate System Transformation Matrices (CSTM) data and material properties (MPT and DIT) data respectively. GMMATS is used by element routines as a general matrix multiply routine and INVERS is used for inversion of small in-core (order usually $\leq$ 12) matrices. It should be noted that all matrices referenced in the structural element subroutines are stored by rows and are single precision. See the subroutine descriptions for these routines in section 3.

The axisymmetric shell element routines STRIR1, STRAP1, STØRD1, STRIR2, STRAP2, STØRD2 utilize the following functions and subroutines whose double precision versions are described in the Module Functional Description for SMA1 (section 4.27.8).

SDR2 (Single precision)	SMA1 (Double precision)
AI	DKI
AK	DKK
AM	DKM
BINT	DKINT
AJ	DKJ
CØEF	DKEF
F89	DK89
FF100	DK100
IFAC	KFAC
FJAB	DKJAB
F6219	DK219
F6211	DK211
RØMBER	RØMBDK
F4	D4K
F5	D5K
F6	D6K
AMATRX	DMATRX

4.46.8.1 Subroutine Name: SDR2AA

1. Entry Point: SDR2AA

2. Purpose: To perform stage I as defined above, under "Method".

3. Calling Sequence: CALL SDR2AA

4. Method: SDR2AA attempts to open the XYCDB data block. If it is purged, a return is given to SDR2. Otherwise, the header record and first data record of XYCDB are skipped and data applying to all subcases are read from the second data record. If no such data exist a dummy master is created. Otherwise, the master data are reduced to a list of unique pairs. If only master data exist, flags are set appropriately.

For each record in the Case Control data block the following processing occurs.

a. The record is read into core. If no XYCDB subcase corresponds to the Case Control subcase, pointers are set to the master data. Otherwise, the master data and appropriate XYCDB subcase data are merged and reduced to unique pairs.

b. For each request for solution set output in XYCDB, the corresponding request in Case Control is examined. If no request is present in Case Control, the XYCDB request is reduced to a set in Case Control format, and a request for the set is turned on in Case Control. If the Case Control set is "ALL", no further action is taken. If the Case Control request is a set, the set is "merged" with the XYCDB set, and the request is altered to reflect the new set (unless all points in the XYCDB set were already in the Case Control set). A flag is set if any new requests are formed.

c. When all requests for the current Case Control record have been analyzed, the record (as modified) is written on a scratch file.

d. When all Case Control records have been read, the GINØ file name for the Case Control data block is switched to the scratch file (unless no modifications were made to Case Control).

4.46.8.2 Subroutine Name: SDR2A

1. Entry Point: SDR2A

2. Purpose: To perform stage II as defined above, under "Method".

3. Calling Sequence: CALL SDR2A

4. Method: SDR2A analyzes Case Control to determine the overall output requests. Flags are set to zero for all request possibilities, and, as each record of Case Control is analyzed, flags are set to 1 for those requests present. Simultaneously, a master set of all structural elements requested for output is created. The master set is then a union of all element requests present with duplicates removed.

4.46.8.3 Subroutine Name: SDR2B

1. Entry Point: SDR2B

2. Purpose: To perform stage III as defined above, under "Method".

3. Calling Sequence: CALL SDR2B

4. Method: SDR2B performs the "phase I" stress and force recovery computations.

The CSTM is first read into core, if present, and then the material property data are read into core via the routine PREMAT. At this point, the EST is opened and processed with one pass.

SDR2A determined a master set list of all elements requested by the user to be output. This master set list, residing in core, is now used as the EST is processed.

For each record of the EST a particular element type is represented. Thus the first word of an EST record (giving the element type) is read, and the element dependent variables are set. The element summary for each element of this type is read, and, if the element is included in the master set, the phase I stress recovery routine is called to compute the element stress matrices which are functions of element geometry and material properties only. The results of this computation are output to a scratch data block for use by SDR2D (stage V). When the end-of-record is encountered on the EST for this element type, the next element type record is processed, until all records of the EST have been passed.

4.46.8.4 Subroutine Name: SRØD1

1. Entry Point: SRØD1
2. Purpose: To generate element stress matrices for the RØD element.
3. Calling Sequence: CALL SRØD1

4.46.8.5 Subroutine Name: SBEAM1

1. Entry Point: SBEAM1
2. Purpose: To generate element stress matrices for the BEAM element.
3. Calling Sequence: CALL SBEAM1

4.46.8.6 Subroutine Name: STUBE1

1. Entry Point: STUBE1
2. Purpose: To generate element stress matrices for the TUBE element.
3. Calling Sequence: CALL STUBE1

4.46.8.7 Subroutine Name: SPANL1

1. Entry Point: SPANL1
2. Purpose: To generate element stress matrices for the SHEAR and TWIST elements.
3. Calling Sequence: CALL SPANL1 (IARG)

IRAG = { 4 implies SHEAR panel element stress matrices will be generated.
5 implies TWIST panel element stress matrices will be generated.

4.46.8.8 Subroutine Name: STRBS1

1. Entry Point: STRBS1
2. Purpose: To generate element stress matrices for the TRBSC element and perform sub-computations for the SQPDL1 and STRPL1 routines.

3. Calling Sequence: CALL STRBS1 (IARG)

IARG { 0 = TRBSC element
       1 = Sub-computations for SQDPL1.
       2 = Sub-computations for STRPL1. }

4.46.8.9 Subroutine Name: STRPL1

1. Entry Point: STRPL1

2. Purpose: To generate element stress matrices for the TRPLT element.

3. Calling Sequence: CALL STRPL1

4.46.8.10 Subroutine Name: SQDPL1

1. Entry Point: SQDPL1

2. Purpose: To generate element stress matrices for the QDPLT element.

3. Calling Sequence: CALL SQDPL1

4.46.8.11 Subroutine Name: STRME1

1. Entry Point: STRME1

2. Purpose: To generate element stress matrices for the TRMEM element and perform sub-computations for the SQDME1 routine.

3. Calling Sequence: CALL STRME1 (IARG)

IARG { 0 = TRMEM
       1 = Sub-computations for SQDME1 subroutine. }

4.46.8.12 Subroutine Name: SQDME1

1. Entry Point: SQDME1

2. Purpose: To generate element stress matrices for the QDMEM element.

3. Calling Sequence: SQDME1

4.46.8.13 Subroutine Name: SELAS1

1. Entry Point: SELAS1

2. Purpose: To generate element stress matrices for the elements listed under the Calling Sequence.

3. Calling Sequence: CALL SELAS1 (IARG)

IARG
- 1 = ELAS1
- 2 = ELAS2
- 3 = ELAS3
- 4 = ELAS4

4.46.8.14 Subroutine Name: STRQD1

1. Entry Point: STRQD1

2. Purpose: To generate element stress matrices for the elements listed under the Calling Sequence.

3. Calling Sequence: CALL STRQD1 (IARG)

IARG
- 1 = TRIA1
- 2 = TRIA2
- 3 = QUAD1
- 4 = QUAD2

4.46.8.15 Subroutine Name: SBAR1

1. Entry Point: SBAR1

2. Purpose: To generate element stress matrices for the BAR element.

3. Calling Sequence: CALL SBAR1

4.46.8.16 Subroutine Name: SCØNE1

1. Entry Point: SCØNE1

2. Purpose: To generate element stress matrices for the CØNE element.

3. Calling Sequence: CALL SCØNE1

4.46.8.17 Subroutine Name: STRIR1

1. Entry Point: STRIR1

2. Purpose: To generate element stress matrices for the TRIRG element.

3. Calling Sequence: CALL STRIR1

4.46.8.18 Subroutine Name: STRAP1

1. Entry Point: STRAP1

2. Purpose: To generate element stress matrices for the TRAPRG element.

3. Calling Sequence: CALL STRAP1

4.46.8.19 Subroutine Name: STØRD1

1. Entry Point: STØRD1

2. Purpose: To generate stress matrices for the TØRDRG element.

3. Calling Sequence: CALL STØRD1

4.46.8.20 Subroutine Name: SDR2C

1. Entry Point: SDR2C

2. Purpose: To perform stage IV as defined above, under "Method".

3. Calling Sequence: CALL SDR2C

4. Method: SDR2C operations are dependent on the Rigid Format being executed. In all cases the second record of EQEXIN or EQDYN is first read into core. An over-all loop of 3 passes is then executed to process the following: pass 1: displacements, velocities, accelerations; pass 2: single-point constraint forces; and pass 3: loads.

For each pass the Case Control data block is opened for input, and the following operations are performed depending on Rigid Format:

a. For eigenvalue problems, a list of eigenvalues and mode numbers is read into core from LAMA or CLAMA.

b. For Differential Stiffness or Buckling phase 1 problems, the first record of

Case Control, which is used in phase 0 of Buckling or Differential Stiffness, is skipped.

c. For frequency or transient response problems, a list of frequencies or times is read into core from PPF or PPT.

At this point a record in Case Control is read, and it is determined if a symmetry sequence of length LSYM is to be output. If it is, the previous LSYM vectors of the UGV data block are unpacked and a linear combination is formed in core. Otherwise, UGV is opened, if not yet opened, and the next vector present is unpacked into core.

Data items are now assembled for the identification record, and this identification record is output to the output data block. Output line entries for the point-ID's requested are then written on the output data block forming a data record. At this time, if the user requested magnitude/phase for complex outputs, the magnitude/phase computations are performed on the real/imaginary pairs.

When all requests have been processed for this vector, the next Case Control record is read. If no more Case Control records exist and there are more vectors present, those vectors are processed using the last Case Control record's specifications.

When all vectors have been processed for the current loop pass, the next pass may be made for forces of single-point constraint or loads.

If deformed structure plots are requested, an output plot data block is formed during the first loop pass, described above, containing translation components of the displacement vector rotated to basic coordinates.

4.46.8.21 Subroutine Name: SDR2D

1. Entry Point: SDR2D
2. Purpose: To perform stage V as defined above, under "Method".
3. Calling Sequence: CALL SDR2D

4. Method: SDR2D performs the phase 2 stress and force recovery computations. In this phase actual stresses and forces are computed for the user-requested elements. These stresses and forces are a function of the stress matrices computed in Stage III and the displacements at the grid points of the elements.

The operations of SDR2D are dependent upon the Rigid Format being executed. In all cases the Case Control data block is opened first. For eigenvalue problems a list of eigenvalues and mode numbers is read into core from LAMA or CLAMA. For Differential Stiffness or Buckling phase 1 problems, the first record of Case Control, which is used in phase 0 of Buckling or Differential Stiffness, is skipped. For frequency or transient response problems, a list of frequencies or times is read into core from PPF or PPT.

Core and GINØ buffers are then allocated as required for a) the Case Control data block, b) the Element Deformation Table, c) the Grid Point Temperature Table, and d) the element stress matrices. If there is insufficient space in core for the element stress matrices, they are maintained on the scratch data block generated in stage III.

The displacement data block (UGV) is now opened, and the displacement vectors present are processed serially with Case Control as in stage IV. Each element requested for output has its respective phase 2 element stress and force recovery routine called. The element routine outputs form the entries for the output data record of this element type. In the case of a complex displacement vector, the element routine is called first with a pointer to the real displacement vector and then with a pointer to the imaginary displacement vector. The results of these two calls are merged to form the complex output stresses and forces.

4.46.8.22 Subroutine Name: SDR2E

1. Entry Point: SDR2E

2. Purpose: To pass through the element stress matrices once, executing the final element stress and force computations for the requests in the current subcase of Case Control.

3. Calling Sequence: SDR2E ($n)

n = FØRTRAN statement number defining the return taken in the event of an error in SDR2E.

4.46.8.23 Subroutine Name: SRØD2

1. Entry Point: SRØD2

2. Purpose: To perform final stress and force computations for the RØD element.

3. Calling Sequence: CALL SRØD2

4.46.8.24 Subroutine Name: SBEAM2

1. Entry Point: SBEAM2

2. Purpose: To perform final stress and force computations for the BEAM element.

3. Calling Sequence: CALL SBEAM2

4.46.8.25 Subroutine Name: SPANL2

1. Entry Point: SPANL2

2. Purpose: To perform final stress and force computations for the SHEAR and TWIST elements.

3. Calling Sequence: CALL SPANL2 (IARG)

IARG { 4 = SHEAR element.
     { 5 = TWIST element.

4.46.8.26 Subroutine Name: SELAS2

1. Entry Point: SELAS2

2. Purpose: To perform final stress and force computations for the ELAS1, ELAS2, ELAS3, and ELAS4 elements.

3. Calling Sequence: CALL SELAS2

4.46.8.27 Subroutine Name: SBSPL2

1. Entry Point: SBSPL2

2. Purpose: To perform final stress and force computations for the TRBSC, TRPLT, and QDPLT elements.

3. Calling Sequence: CALL SBSPL2 (IARG)

IARG { 0 = TRBSC element.
       3 = TRPLT element.
       4 = QDPLT element.

4.46.8.28 Subroutine Name: STQME2

1. Entry Point: STQME2

2. Purpose: To perform final stress computations for the TRMEM and QDMEM elements.

3. Calling Sequence: CALL STQME2 (IARG)

IARG { 1 = TRMEM element.
       2 = QDMEM element.

4.46.8.29 Subroutine Name: STRQD2

1. Entry Point: STRQD2

2. Purpose: To perform final stress and force computations for the TRIA1, TRIA2, QUAD1, and QUAD2 elements.

3. Calling Sequence: CALL STRQD2 (IARG)

IARG { 3 = TRIA1 or TRIA2 element.
       4 = QUAD1 or QUAD2 element.

4.46.8.30 Subroutine Name: SCØNE2

1. Entry Point: SCØNE2

2. Purpose: To perform final harmonic stress and force computations for the CØNE element.

3. Calling Sequence: CALL SCØNE2

4.46.8.31 Subroutine Name: SCØNE3

1. Entry Point: SCØNE3

2. Purpose: To compute the final stresses and forces for one of the 14 possible points in a CØNE element.

3. Calling Sequence: CALL SCØNE3 (LARG)

LARG = Logical argument set .FALSE. initially and then set .TRUE. by SCØNE3 after the last point defined for a particular element has had its final stresses and forces computed.

4.46.8.32 Subroutine Name: SBAR2

1. Entry Point: SBAR2

2. Purpose: To perform final stress and force computations for the BAR element.

3. Calling Sequence: CALL SBAR2

4.46.8.33 Subroutine Name: STRIR2

1. Entry Point: STRIR2

2. Purpose: To perform final stress and force computations for the TRIRG element.

3. Calling Sequence: CALL STRIR2 (TGRID)

TGRID = 3 word real array giving grid point temperatures at the 3 connection grid points.

4.46.8.34 Subroutine Name: STRAP2

1. Entry Point: STRAP2

2. Purpose: To perform final stress and force computations for the TRAPRG element.

3. Calling Sequence: CALL STRAP2 (TGRID)

TGRID = 4 word real array giving grid point temperatures at the 4 connection grid points.

4.46.8.35 Subroutine Name: STØRD2

1. Entry Point: STØRD2

2. Purpose: To perform final stress and force computations for the TØRDRG element.

3. Calling Sequence: CALL STØRD2 (TGRID)

TGRID = Two-word real array giving grid point temperatures at the two connection grid points.

4.46.8.36 Subroutine Name: MAGPHA

1. Entry Point: MAGPHA

2. Purpose: To compute the magnitude and phase of a complex number, c.

3. Calling Sequence: CALL MAGPHA (A,B)

A - Real part of c on input, magnitude of c on return.

B - Imaginary part of c on input, phase of c on return.

4.46.9 Design Requirements

1. Since the five stages of the SDR2 module must be able to operate under all Rigid Formats, a branch design has been used within each stage whereby operations that are variant under different Rigid Formats are grouped into substructures within that stage. At these locations, a branch is always made to the substructure appropriate to the Rigid Format being executed.

2. The following common blocks appear only within the subroutines of the SDR2 module.

a. CØMMØN/SDR2X1/

This common block contains seventeen flag words which are set on the basis of requests in the Case Control data block by SDR2A.

b. CØMMØN/SDR2X2/

This common block contains twenty-nine pointers defining the locations of the various requests and set definitions within the Case Control data block.

c. CØMMØN/SDR2X3/

This common block contains data constants unique to the structural elements.

d. CØMMØN/SDR2X4/

This common block contains local variables and flags set by the subroutines of the

SDR2 module for communication between these subroutines.

e. CØMMØN/SDR2X5/

This common block is used by SDR2B to send EST data to the phase 1 element routines and to receive outputs from the phase 1 element routines.

f. CØMMØN/SDR2X6/

This common block consists of a three hundred word scratch area for use by the element routines while performing phase 1 computations.

g. CØMMØN/SDR2X7/

This common block is used by SDR2D for sending and receiving data to the element routines performing phase 2 computations.

h. CØMMØN/SDR2X8/

This common block consists of a three hundred word scratch area in core for use by the element routines while performing phase 2 computations.

4.46.10 Diagnostic Messages

SDR2 being one of the last modules to execute in a problem solution, makes every attempt at execution in any event. Should an error be detected by SDR2, a message is queued for output and, if possible, SDR2 continues to execute portions of the solution not affected by the error. The following NASTRAN messages may be output by SDR2: 2075, 2076, 2077, 2078, 2079, 2080, 3001, 3002, 3003, and 3008.

4.47 FUNCTIONAL MODULE DPD (DYNAMICS POOL DISTRIBUTOR)

4.47.1 Entry Point: DPD

4.47.2 Purpose

DPD is the principal data processing module for dynamics problems. New tables are assembled to account for any extra points in the model and the additional displacement sets used in dynamics. Bulk data cards which control the solution of a dynamics problem are processed and assembled into various data blocks for convenience and efficiency in solution of the dynamics problem.

4.47.3 DMAP Calling Sequence

```
DPD     DYNAMICS,GPL,SIL,USET/GPLD,SILD,USETD,TFPØØL,DLT,PSDL,FRL,NLFT,TRL,EED,EQDYN/
        V,N,LUSET/V,N,LUSETD/V,N,NØTFL/V,N,NØDLT/V,N,NØPSDL/V,N,NØFRL/V,N,NØNLFT/
        V,N,NØTRL/V,N,NØEED/C,N,O/V,N,NØUE   $
```

4.47.4 Input Data Blocks

DYNAMICS - Collection of bulk data cards for dynamics problem.

GPL - Grid Point List.

SIL - Scalar Index List.

USET - Displacement set definitions table.

Note: DYNAMICS may be purged. Other input data blocks may not be purged.

4.47.5 Output Data Blocks

GPLD - Grid Point List Dynamics.

SILD - Scalar Index List Dynamics.

USETD - Displacement set definition table dynamics.

TFPØØL - Transfer Function Pool.

DLT - Dynamic Loads Table.

PSDL - Power Spectral Density List.

FRL - Frequency Response List.

NLFT - Non-Linear Forcing Table.

TRL - Transient Response List.

EED - Eigenvalue Extraction Data.

EQDYN - Equivalence between external and internal numbers - dynamics.

Note: GPLD, SILD, USETD and EQDYN may not be purged. All other data blocks may be purged.

4.47.6 Parameters

LUSET - Input-integer-no default. Degrees of freedom in the g - displacement set.

LUSETD - Output-integer-no default. Degrees of freedom in the p - displacement set.

NØTFL - Output-integer-no default. Number of transfer function sets in the bulk data, -1 if no sets are defined.

NØDLT - Output-integer-no default. +1 if dynamics load data are present in the bulk data (i.e., DLT is created), -1 otherwise.

NØPSDL - Output-integer-no default. +1 if the PSDL is created, -1 otherwise.

NØFRL - Output-integer-no default. +1 if the FRL is created, -1 otherwise.

NØNLFT - Output-integer-no default. +1 if the NLFT is created, -1 otherwise.

NØTRL - Output-integer-no default. +1 if the TRL is created, -1 otherwise.

NØEED - Output-integer-no default. +1 if the EED is created, -1 otherwise.

NØUE - Output-integer-no default. Number of extra points in the model, -1 if there are no extra points.

4.47.7 Method

4.47.7.1 General

Subroutine DPD is the main control program for the module. It initializes each of the DMAP parameters, allocates buffers in open core (/DPDCØR/), and calls each of the principal routines of the module as follows.

1. DPD1 to assemble GPLD, USETD, SILD and EQDYN.
2. DPD2 to assemble DLT.
3. DPD3 to assemble FRL and PSDL.
4. DPD4 to assemble NLFT and TRL.
5. DPD5 to assemble EED and TFL.

4.47.7.2 Assembly of GPLD, USETD, SILD and EQDYN

The second logical record of GPL, which contains pairs of external point identification and sequence numbers, is read into core. Three words are used for each entry in the GPL. In the third word of each entry the internal index is stored. The list of extra points is read from the EPØINT record in DYNAMICS. For each extra point, a three-word entry is added to the list now in core. The first word contains the extra point identification, the second contains the initial sequence number equal to 1000 times the point ID, and the third word is zero. The SEQEP record in DYNAMICS is read. For each referenced point, the sequence number is replaced by the new sequence number. The list in core is now sorted on sequence number by subroutine SØRT. The sequence number in each entry is replaced by an internal index according to the position of the entry following the sort. The GPLD is now written. It consists of one logical record of one word entries, each entry containing the external point identification. The internal index is implied by the position of the entry in the record.

The SIL is read into core following the table of three-word entries currently in core. Bit masks are initialized for the various displacement sets in statics and dynamics. Files containing the USETD and SILD data blocks are opened to write. The file containing the USET data block is opened to read. Each of the three-word entries in core is processed as follows.

1. If the entry corresponds to a grid point, six words are read from USET, bits for displacement sets in dynamics are turned on according to the statics sets to which the point belongs, six words are written on USETD, one word is written on SILD, the second word of the three-word entry is replaced with the scalar index value in the p-set, the third word is replaced with the scalar index value in the g-set, and the scalar index counter for the p-set is incremented by six.

2. If the entry corresponds to a scalar point, one word is read from USET, and the process proceeds as above except that one word is written on USETD and the scalar index counter is incremented by one.

3. If the entry corresponds to an extra point, one word is written on USETD, the extra point counter is incremented, and the process proceeds as with a scalar point.

When all entries have been processed, USETD is complete and the first logical record of SILD is complete. The second logical record of SILD is now written. It comprises two-word entries, each pair containing a scalar index value in the g-set and the corresponding scalar index value in the p-set.

The third word of each of the three word entries in core is now replaced with a code word which is ten times the scalar index value in the p-set plus the type of point (1 = grid, 2 = scalar, 3 = extra). The table is sorted on external point identification. EQDYN is written as two logical records. The first record contains pairs, each consisting of an external point ID and a scalar index value in the p-set. The second record contains pairs, each consisting of an external point ID and a code word.

4.47.7.3 Assembly of the DLT

The DAREA, DELAY and DPHASE tables are read from DYNAMICS, one table at a time. Grid point and component codes are converted to a scalar index value in the p-set by subroutine DPDAA. When all entries of a table have been read, the table is sorted on scalar index value and written as a logical record on a scratch file (three scratch files are used, one for each of the three types of tables). The table identification is saved in core.

The RLØAD1, RLØAD2, TLØAD1 and TLØAD2 cards are read from DYNAMICS and stored in core. Eleven words are used for each entry. In the first word of each entry, a code for the card type is stored. When all cards have been read and stored in core, the data are sorted on load set identification.

DLØAD cards are read from DYNAMICS and stored in core, and the data within each DLØAD card are sorted on referenced set identification. The file containing the DLT data block is opened to write. The header record is written. It contains the data block name, a list of set

identifications defined on DLØAD cards and a list of set identifications defined on RLØAD1, RLØAD2, TLØAD1 and TLØAD2 cards. The DLØAD data are then written as the first logical record of the DLT. The remainder of the DLT comprises one logical record per load set. For each entry in the 11-word per entry table in core the following processing occurs.

1. The scratch file containing the tables referenced in the entry is positioned to read the referenced table.

2. Entries are read from each of the referenced tables. A four-word entry is written on the DLT consisting of the scalar index value, A, τ, and Θ.

3. When all entries from the tables have been read, the DLT record is closed, and the process repeats for the next load set.

4.47.7.4 Assembly of the FRL and the PSDL

FREQ1, FREQ2 and FREQ cards are read from DYNAMICS and stored in core. Frequencies on FREQ cards are converted to radians. When all the data have been read, a list containing three words per entry is accumulated in core. The first word contains the frequency set identification, the second word contains a pointer to the first word where data belonging to the set are stored, and the third word defines the type (0 = FREQ1, -1 = FREQ2, N = FREQ, where N is the number of words in the frequency set). The list is sorted on set identification. The file containing the FRL data block is opened to write. The header record is written. It contains the data block name and a list of all frequency set identifications. The remainder of the FRL is comprised of one logical record per frequency set.

For each entry in the three-word per entry list in core the following processing occurs.

1. If the entry corresponds to a FREQ1 set, then N+1 frequencies are written as a logical record on the FRL by the following equation:

$$f_i = f_0 + (i-1)\,\Delta f,\ i = 1,2,\ldots,N+1. \qquad (1)$$

2. If the entry corresponds to a FREQ2 set, then N+1 frequencies are written as one logical record on the FRL by the following equation:

$$f_i = f_o \quad 10^{(i-1)\delta} \quad i = 1,2,\ldots, N+1 \tag{2}$$

where

$$\delta = \frac{1}{N} \log_{10} \left(\frac{f_e}{f_o}\right). \tag{3}$$

3. If the entry corresponds to a FREQ set, the frequencies in the set are sorted, and any duplicate frequencies are discarded. The sorted list is written as one logical record on the FRL.

The RANDPS cards are read into core (if no RANDPS data are present, the PSDL is not assembled). The RANDT1 and RANDT2 cards are read into core, and a list similar to that in the frequency processing is formed. This list is sorted on set identification number. The file containing the PSDL is opened to write, and the set identifications are written in the header record. The RANDPS data are written as the first logical record of the PSDL. The remainder of the PSDL contains one logical record per set. For RANDT2 sets, the data are sorted on time lag, and duplicates are discarded prior to writing the record. For RANDT1 sets, N + 1 time lags, τ_i, are written where

$$\tau_i = \tau_o + (i-1)\Delta\tau \quad i = 1,2,\ldots, N+1 \tag{4}$$

4.47.7.5 Assembly of the NLFT and TRL

The NØLINi (i = 1,2,3,4) cards are read into core. Each referenced grid point and component code is converted to a scalar index value in the u_p-set. The data are sorted on set identification number. USETD is read into core. The file containing the NLFT data block is opened to write, and the set identifications are written in the header record. The remainder of the NLFT contains one logical record per set. Scalar index values within each set are converted to scalar index values in the u_d and u_e sets. The data within each set are sorted on the scalar index value to which the forcing function is applied.

The TIC cards are read, referenced grid points and component codes are converted to scalar index values in the u_p-set, and the data are written on SCR1, one logical record per set. A list of the TIC set identifications is accumulated in core. USETD is read into core. The

file containing the TRL data block is opened to write. The set identifications are written in the header record. The last word of the header contains the degrees of freedom in the u_d-set. Data are read from SCR1. Scalar index values are converted to scalar index values in the u_d-set. Each TIC set is written as one logical record on the TRL. When all the TIC data have been processed, the TSTEP data are copied from DYNAMICS to the TRL, one logical record per TSTEP set.

4.47.7.6 Assembly of the EED and TFL

Processing of EIGB, EIGC, EIGP and EIGR cards is minimal. For each card type present, a corresponding logical record is written on EED. For each of the cards which specify PØINT, the referenced grid point and component code is converted to a scalar index value (u_a set for EIGB and EIGR cards, u_d set for EIGC cards).

Transfer function data are read from the TF record on DYNAMICS one set at a time. For each transfer function set, the point and component codes are converted to scalar index values in the u_p set, which in turn form row and column numbers of the transfer function matrices. The data are written on the TFL, one transfer function set per logical record. The set identification number is the first word of each logical record. Four word entries follow. The first word is 65536*column number plus row number; the next three words are the terms of the matrices.

4.47.8 Subroutines

Auxiliary subroutines DPD1, DPD2, DPD3, DP4 are described above.

4.47.8.1 Subroutine Name: DPDAA

1. Entry Point: DPDAA

2. Purpose: To convert a grid point and component code to a scalar index value in the u_p set.

3. Calling Sequence: CALL DPDAA

4. Method: A flag called INEQ is maintained in /DPDCØM/. If the flag is zero, EQDYN is read into core and INEQ is set to one. The grid point and component to be converted is stored in BUF(L) and BUF(L+1) where BUF and L are in /DPDCØM/. A binary search is performed in EQDYN. If the point is found, the corresponding scalar index value is stored in BUF(L). Otherwise, an error message is queued, and an internal

NØGØ flag is turned on.

4.47.9 Design Requirements

4.47.9.1 Allocation of Core Storage

In general, core storage requirements in DPD are the EQDYN table plus one set of data being processed plus two or three GINØ buffers. In DPD3 where EQDYN is not required, it is assumed that all data required to assemble the FRL or PSDL can be held in core at one time.

4.47.9.2 Environment

The Block Data program DPDCBD initializes /DPDCØM/ with GINØ file names, LØCATE codes for the various card types processed by DPD, and miscellaneous data. It must be resident in core when DPD is executed.

DPD is designed to operate in a single overlay segment. Communication in the module occurs through /DPDCØM/ and open core /DPDCØR/. If an alternate overlay is desired, DPDCDB, DPD and DPDAA could form a local primary segment and each of DPD1, DPD2, DPD3, DPD4 and DPD5 could form separate secondary segments. In this case, /DPDCØR/ must be inserted after the longest of the secondary segments. Four scratch files are used by DPD.

4.47.10 Diagnostic Messages

The following messages may be issued by DPD:

2064, 2066, 2068, 2069, 2071, 2107, 2135, 2136.

4.48 FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)

4.48.1 Entry Point: REIG

4.48.2 Purpose:

To solve the equation

$$\Big([K] - \lambda [M]\Big)\{u\} = 0 \qquad (1)$$

for eigenvalues λ and their associated eigenvectors.

4.48.3 DMAP Calling Sequence

```
READ      KAA,MAA,MR,DM,EED,USET,CASECC/LAMA,PHIA,MI,ØEIGS/V,N,FØRMAT/V,N,NEIGVS/V,N,NSKIP  $
```

4.48.4 Input Data Blocks

KAA - Partition of stiffness matrix - a set.

MAA - Partition of mass matrix - a set.

MR - Rigid body mass matrix - r set.

DM - Rigid body transformation matrix.

EED - Eigenvalue Extraction Data.

USET - Displacement set definitions table.

CASECC - Case Control Data Table.

Notes:

1. KAA must be present.
2. MR may or may not be present.
3. DM and USET must be present if MR is present.
4. EED and CASECC must be present.

4.48.5 Output Data Blocks

LAMA - Real Eigenvalue Table.

PHIA - Eigenvectors matrix giving the eigenvectors in the a set.

MI - Modal Mass Matrix.

ØEIGS - Real Eigenvalue Summary Table.

4.48.6 Parameters

FØRMAT - Input-BCD-no default. If FØRMAT ≠ MODES, READ will solve a buckling problem (i.e., $[\lambda^2 M - K] \{u\} = 0$) using EIGB data cards where M is the differential stiffness matrix.

NEIGVS - Output-integer-no default. NEIGVS is the number of eigenvalues found. If none were found, NEIGVS = -1.

NSKIP - Input-integer-no detault. The method used by READ is taken from the NSKIP+1 record of CASECC.

4.48.7 Method

REIG is the main controlling program for the READ module. Its responsibility is to decide which method was asked for (Inverse Power, Determinant or Givens) and to pass control to the appropriate routine. Once eigenvalues have been extracted, REIG directs the sorting and normalizing of the vectors for final output. The flow of the module can be seen in the flow charts shown in Figure 1.

4.48.8 Subroutines

The subroutines used by READ are divided into five classes: 1) subroutines used by REIG, 2) subroutines used for the Inverse Power Method, 3) subroutines used for the Determinant Method, 4) subroutines used for the Givens Method, and 5) general subroutines. The descriptions for the general subroutines can be found in section 3.

FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)

REIG	INVERSE POWER	DETERMINANT	GIVENS
READ1	INVPWR	DETM SUMM	VALVEC
READ2	INVP1	DETM1 SQRTM	SMLEIG
READ3	INVP2	DETM2 DETFBS	TRIDI
READ4	INVP3	DETM3	SICØX
READ5	NØRM1	DETM4	SINCAS
READ6	MTIMSU	DETM5	RØTAX
ØRTCK	XTRNSY	DETM6	RØTATE
INVERT	SUB	FDVECT	EMPCØR
INVTR	INVFBS	EADD	FILCØR
		DETDET	QRITER
		ARRM	WILVEC

GENERAL

DECØMP

ADD

PRELØC

SSG2B

SDR1B

SDCØMP

FACTØR

TRANP1

TRNSP

MERGE

MPYAD

4.48.8.1 Subroutine Name: READ1

1. Entry Point: READ1

2. Purpose: To compute the eigenvectors for the rigid body modes.

3. Calling Sequence: CALL READ1 (DM,MR,SCR1,SCR2,SCR3,PHIAT,USET,NR,LAMAT,SCR4)

DM,MR,USET are the GINØ file numbers of their respective data blocks - integer - input.

SCR1,...,SCR4 are the GINØ file numbers of 4 scratch files - integer - input.

PHIAT - GINØ file number of a temporary storage file for the eigenvectors - integer - input.

LAMAT - GINØ file number of a temporary storage file for the eigenvalues - integer - input.

NR - Number of rigid body modes - integer - output.

CØMMØN /READ1A/Z(1)

Z(1) - Array of open core for READ1

4. Method: Let r be the number of rigid body modes and let

$$I_1 = \begin{Bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{Bmatrix}, \tag{2}$$

$$I_2 = \begin{Bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{Bmatrix}, \tag{3}$$

etc.

Set $\{v_1\} = \{I_1\}$ and perform the following three steps for i = 1, 2,...,r.

a. Normalize $\{v_i\}$ by the following equations:

$$S_i = \{v_i\}^T [m_r] \{v_i\}, \tag{4}$$

$$\{\Phi_i\} = \frac{1}{\sqrt{S_i}} \{v_i\}. \tag{5}$$

S_i must be greater than zero for a consistent rigid body system.

b. Calculate using $j = 1,2,\ldots,i$:

$$\alpha_j = \{\Phi_j\}^T [m_r] \{I_{i+1}\}. \tag{6}$$

c. The next vector is then:

$$\{v_{i+1}\} = \{I_{i+1}\} - \sum_{j=1}^{i} \alpha_j \{\Phi_j\} . \tag{7}$$

(Return to step b).

This procedure is a modification of the Schmidt orthogonalization procedure using the {I} vectors as a starting point. Since the $[m_r]$ matrix is non-singular, the {I} vectors are independent with respect to the matrix. Each new vector $\{I_{i+1}\}$ is made orthogonal with respect to the previous vectors by subtracting its scalar matrix products (α) with the other vectors. The matrix of resulting vectors $[\Phi_{ro}]$ should form a diagonal, unit matrix $[m_{ro}]$ with the equation:

$$[m_{ro}] = [\Phi_{ro}]^T [m_r] [\Phi_{ro}] . \tag{8}$$

The remaining displacements of the rigid body eigenvectors are formed from the equation

$$[\Phi_{\ell o}] = [D] [\Phi_{ro}], \tag{9}$$

where [D] corresponds to the data block DM.

Each column of $[\Phi_{ro}]$ is merged with $[\Phi_{\ell o}]$ using USET (UA,UL,UR).

4.48.8.2 Subroutine Name: READ2

1. Entry Point: READ2

2. Purpose: To compute the modal mass matrix [MI], to normalize the extracted eigenvalues, and to prepare the output files LAMA and ØEIGS.

3. Calling Sequence: CALL READ2 (MAA,PHIA,SCR1,NØRM,IA,USET,MI,LAMA,ØEIGS,SCR2,EPSI,SCR3)

MAA,PHIA,USET,MI,LAMA,ØEIGS are the GINØ file numbers of their respective data blocks - integer - input.

SCR1,...,SCR3 are GINØ file number of 3 scratch files - integer - input.

NØRM - Normalization method requested.

- MAX - Implies maximum component.
- PØINT - Implies specified component.
- MASS - Implies unit modal mass matrix.

} Input - BCD

IA - If NØRM = PØINT, IA is the component number which is set to 1.0 - integer - input.

EPSI - If EPSI ≠ 0.0, the off-diagonal terms of the modal mass matrix [MI] are checked for the number which exceed EPSI.

CØMMØN /READ2A/Z(1)

Z(1) - Array of open core for READ2.

4.48.8.3 Subroutine Name: READ3

1. Entry Point: READ3

2. Purpose: To sort the eigenvalues in ascending order and to output the eigenvalues and eigenvectors in order. Also, to pack the eigenvectors in standard matrix format.

3. Calling Sequence: CALL READ3 (NØVECT,NCØL,SCR1,SCR2,PHI,LAMBDA)

NØVECT - Number of eigenvectors extracted - integer - input.

NCØL - Length of the vectors - integer - input.

SCR1 - GINØ file containing the unsorted eigenvalues - integer - input.

SCR2 - GINØ file containing the unsorted eigenvectors - integer - input.

PHI - GINØ file for the output sorted vectors - integers - input.

LAMBDA - GINØ file for the output sorted eigenvalues - integer - input.

CØMMØN /READ2A/Z(1)

Z(1) - Area of open core available to READ3.

4.48.8.4 Subroutine Name: READ4

1. Entry Point: READ4

2. Purpose: To test for close and equal roots found by the Determinant Method and make sure the corresponding vectors are orthogonal.

3. Calling Sequence: CALL READ4 (LAMAT,MPHIA,SCR1,EPSI,MAA)

LAMAT - GINØ file number of temporary storage file for the eigenvalues found - integer - input.

MPHIA - Matrix control block for PHIA - integer - input.

SCR1 - GINØ file number of a scratch file - integer - input.

EPSI - Close root test criteria - real - input.

MAA - GINØ file number of MAA - integer - input.

CØMMØN /READ2A/Z(1) - See READ2A

Z(1) - Array of open core for READ4.

4.48.8.5 Subroutine Name: ØRTCK

1. Entry Point: ØRTCK

2. Purpose: ØRTCK will generate the generalized mass matrix for the close roots and make the epsilon test to determine if the vectors should be orthogonalized.

3. Calling Sequence: CALL ØRTCK (X,MAA,BUFFER(1),NUM,NDIM,GM,ACCUM,EPSI)

X - Unorthogonalized eigenvectors - real - input/output.

MAA and EPSI are as described in READ4.

BUFFER - GINØ buffer.

NUM - Number of close roots - integer - input.

NDIM - Order of the problem - integer - input.

GM - Generalized mass for the close roots - real array - output.

ACCUM - Running sum of $[M_{aa}]$ $[\Phi_a]$ - real - input/output.

4.48.8.6 Subroutine Name: INVPWR

1. Entry Point: INVPWR

2. Purpose: INVPWR is the main driver for the Inverse Power Method of eigenvalue extraction.

3. Calling Sequence: CALL INVPWR

CØMMØN /INVPWX/K(7),M(7),LAM(7),PHI(7),SCRFIL(8),EIGSUM,LMIN,LMAX,NØEST,NDPLUS,NDMNUS, EPS,NØRTHØ

K,M - Matrix control blocks for the input stiffness and mass matrices, [K] and [M].

LAM,PHI - Matrix control blocks for the output eigenvalue and eigenvector files.

SCRFIL(8) - GINØ file numbers for eight scratch files.

EIGSUM - GINØ file number for the eigenvalue summary file - integer.

LMIN-LMAX - Desired range for eigenvalues - real.

NØEST - Number of estimated eigenvalues in the range - integer.

NDPLUS - Number of desired positive eigenvalues - integer.

NDMNUS - Number of desired negative eigenvalues - integer.

EPS - Convergence criteria - real.

NØRTHØ - Number of roots extracted - integer.

CØMMØN /INVPX/Z(1)

Z(1) - Area of open core available to INVPWR.

4. Method: Reference can be made to the flow charts in Figure 1 for the program logic. Theoretical implications are handled in section 4.48.11 and in the Theoretical Manual.

4.48.8.7 Subroutine Name: INVP1

1. Entry Point: INVP1

2. Purpose: To set up a call to ADD to form

$$[A] = [K] - \lambda_o [M] \tag{10}$$

3. Calling Sequence: CALL INVP1

CØMMØN /INVPWX/K(7),M(7),A

CØMMØN /INVPXX/LAMØ

CØMMØN /INVP1X/Z(1)

K,M - Matrix control blocks for the input matrices.

A - GINØ file number for the output matrix.

LAMØ - Double precision scalar multiplier.

Z(1) - Area of open core available to ADD.

4.48.8.8 Subroutine Name: INVP2

1. Entry Point: INVP2

2. Purpose: To initialize and call DECØMP for subroutine INVPWR.

3. Calling Sequence: CALL INVP2

CØMMØN /INVPWX/DUM(14),A(7),L(7),XX(2),U,SCR1,SCR2,SCR3,LL,UU

CØMMØN /INVPXX/DUMM(12),SWITCH

CØMMØN /INVP2X/Z(1)

A - GINØ file number of the input matrix - integer - input.

L,U - GINØ file number for the lower and upper triangular matrices output from DECØMP - integer - input.

SCR1, SCR2, SCR3 } - Three scratch files used by DECØMP - integer - input.

LL,UU - GINØ file numbers for alternate storage of L and U - integer - input.

SWITCH - 0 - Store output matrices in L and U. 1 - Store output matrices in LL and UU - integer - input.

Z(1) - Area of open core available for DECØMP.

4.48.8.9 Subroutine Name: INVP3

1. Entry Point: INVP3

2. Purpose: To solve for an eigenvalue and eigenvector using the Inverse Power Method.

3. Calling Sequence: CALL INVP3

CØMMØN /INVPWX/K(7),M(7),LAM(7),PHI(7),SCRFIL(8),EIGSUM,LMIN,LMAX,NØEST,NDPLUS,NDMNUS,EPS

CØMMØN /INVP3X/Z(1)

See INVPWR for a description of /INVPWX/(section 4.48.8.6).

Z(1) - Open core for INVP3.

4. Method: The logic flow of INVP3 is given in Figure 1 with the mathematical equations supplied in 4.48.11. Theoretical considerations are given in the Theoretical Manual.

INVP3 was designed with two aspects in mind: first to assure that all roots within a given region are found, and second to avoid any possibility of looping uncontrollably. To accomplish these ends, considerable testing was inserted around the mathematical equations.

5. Design Requirements: INVP3 needs sufficient storage to hold seven double precision a set vectors and three GINØ buffers in core.

4.48.8.10 Subroutine Name: NØRM1

1. Entry Point: NØRM1

2. Purpose: To normalize a vector {x} such that its maximum component is one.

3. Calling Sequence: CALL NØRM1 (X,DIV)

CØMMØN /INVPWX/XX,N

N - Length of the double precision vector {x}.

X - Double precision vector {x}.

DIV - Double precision value of the divisor used to normalize {x}- output.

4.48.8.11 Subroutine Name: MTIMSU

1. Entry Point: MTIMSU

2. Purpose: To pre-multiply a vector by a matrix i.e.,:

$$\{x\} = [M]\{y\}. \tag{11}$$

3. Calling Sequence: CALL MTIMSU (Y,X,BUF)

CØMMØN /INVPWX/DUM(7),M(7)

CØMMØN /INVPXX/DUMM(13),NZERØ

M - Matrix control block for matrix [M].

X,Y - Double precision left and right hand side vectors.

BUF - GINØ buffer.

NZERØ - Number of zero columns of matrix [M].

4.48.8.12 Subroutine Name: XTRNSY

1. Entry Point: XTRNSY

2. Purpose: To form the dot product of two vectors

$$a = \{x\}^T \{y\}. \tag{12}$$

3. Calling Sequence: CALL XTRNSY (X,Y,A)

CØMMØN /INVPWX/XX,N

N - Length of vectors {x} and {y}.

X,Y - Double precision vectors.

A - Double precision value of the dot product.

4.48.8.13 Subroutine Name: SUB

1. Entry Point: SUB

2. Purpose: To evaluate the vector equation

$$\{z\} = a\{x\} - b\{y\}. \tag{13}$$

3. Calling Sequence: CALL SUB (X,Y,A,B)

CØMMØN /INVPWX/XX,N

N - Length of the vectors {x} and {y}.

X,Y - Double precision vectors in the above equation. Y, upon return from SUB, contains the difference vector {z}.

A,B - Double precision scalar multipliers.

4.48.8.14 Subroutine Name: INVFBS

1. Entry Point: INVFBS

2. Purpose: INVFBS will perform the forward-backward substitution necessary to solve one iteration of the Inverse Power Method given by

$$([K] - \lambda_o [M])\{W_n\} = [M]\{u_{n-1}\}. \tag{14}$$

3. Calling Sequence: CALL INVFBS (X,Y,BUF)

COMMON /INFBSX/L(7),U(7)

L,U - Matrix control blocks for the lower and upper triangular matrices generated by decomposition of $([K] - \lambda_o [M])$.

X - Double precision right hand vector, $[M]\{u_{n-1}\}$.

Y - Double precision solution vector, $\{W_n\}$.

BUF - GINO buffer.

4. Method: INVFBS is a stripped down version of GFBS. Both vectors are stored in core, and only real double precision arithmetic is performed.

4.48.8.15 Subroutine Name: DETM

1. Entry Point: DETM

2. Purpose: To supervise the operations of the Determinant Method.

3. Calling Sequence: CALL DETM

All determinant routines use common blocks /REGEAN/ and /DETMX/.

CØMMØN /REGEAN/IM(7),IK(7),IEV(7),SCR1,SCR2,SCR3,SCR4,SCR5,LCØRE,RMAX,RMIN,MZ,NEV,EPSI, RMINR,NE,NIT,NEVM,SCR6,SCR7,NFØUND,LAMA,IBUCK,NSYM

IM(7) - Matrix control block for MAA.

IK(7) - Matrix control block for KAA.

IEV(7) - Matrix control block for the eigenvectors.

SCR1,...,
SCR7 - GINØ file number of 7 scratch files - integer - input.

SCR1 - Contains [KAA] - λ[MAA].

SCR2 - Contains [LLL].

SCR3 - Contains [ULL].

LCØRE - Amount of open core reserved for starting points, determinants, scale factors and accepted eigenvalues - integer - input.

RMAX - Maximum eigenvalue of interest - real - input/output.

RMIN - Minimum eigenvalue of interest - real - input.

MZ - Number of zero eigenvalues - integer - input.

NEV - Number of estimated eigenvalues in the range of interest - integer - input.

EPSI - Convergence criteria - real - input/output. EPSI = 1.0×10^{-11} currently.

RMINR - Lower boundary to search region. (RMINR = -.01*RMIN)
For buckling problems a pole of geometrically increasing order is placed at RMINR each time the search procedure points below RMIN - real - input.

NE - Number of changes in the convergence criteria - integer - input. NE is currently set at 4.

NIT - Number of iterations allowed to converge to an eigenvalue. NIT is currently set to 20 - integer - input.

NEVM - Number of eigenvalues desired - integer - input.

NFØUND - Number of eigenvalues found - integer - output.

LAMA - GINØ file number of eigenvalue storage file - integer - input.

IBUCK - Buckling flag. If IBUCK = 3, the current problem is a Buckling Analysis problem - integer - input.

NSYM - Symmetric determinant flag. If NSYM = 1, symmetric decomposition is used to compute the determinant of A - integer - input.

CØMMØN/DETMX/P(4),DETX(4),PS1(4),DET1(4),N2EV,IPSAV,IPS,IDET,IPDETA,PREC,NSTART,NDCMP,IC,NSMØVE
ITERM,IS,ND,IADD,SML1,IPDETX(4),IPDET1(4),IFAIL,K,FACT1,IFFND,NFAIL,NPØLE,ISNG

P - The 3 trial values of the eigenvalue - double precision - input/output.

DETX - The determinants of the 3 P's above - double precision - input/output.

PS1 - The 3 current starting points - double precision - input/output.

DET1 - The determinants of the 3 starting points - double precision - input/output.

N2EV - The number of starting points - integer - input.

IPSAV - Pointer to the accepted eigenvalues - integer - input.

IPS - Pointer to the starting points - integer - input.

IDET - Pointer to the determinants of the starting points - integer - input.

IPDETA - Pointer to the scale factors of the determinants - integer - input.

PREC - Precision of the calculations - integer - input.

NSTART - Number of passes through the starting points - integer - output.

NDCMP - Total number of decompositions - integer - output.

IC - Total number of convergence criteria changes - integer - output.

NSMØVE - Number of starting point moves - integer - output.

ITERM - Reason for termination - integer - output.

IS - Start set counter - integer - input/output.

ND - Number of new starting points to evaluate - integer - input.

IADD - Pointer to next starting point to evaluate - integer - input/output.

SML1 - Magnitude of smallest diagonal element of [ULL] - real - output.

IPDETX - Scale factors for the determinants in DETX - integer - input/output.

IPDET1 - Scale factors for the determinants in DET1 - integer - input/output.

IFAIL - If this set of starting points produced a failure to iterate to a root, IFAIL = 1 - integer - input/output.

K - Current iteration counter - integer - input.

FACT1 - Constant = EPSI*SQRT(RMAX) - real - input/output.

IFFND - If an eigenvalue is accepted on a given pass through the starting points, IFFND = 1 - integer - input/output.

NFAIL - Number of failures to iterate to a root - integer - output.

NPØLE - Order of the current pole at RMINR - integer - input/output.

ISNG - Number of singular matrices detected during decomposition - integer - input/output.

CØMMØN/DETDX/DZ(1)

DZ(1) - Array of open core for DETM.

4. Method: The general method used for the Determinant Method is described in section 4.88.

5. Design Requirements: DETM needs sufficient open core to hold two double precision a set vectors and one GINØ buffer. Open core at /DETDX/ is used for storage of starting points, determinants, powers and accepted eigenvalues.

4.48.8.16 Subroutine Name: DETM1

1. Entry Point: DETM1

2. Purpose: To compute the locations of the starting points.

3. Calling Sequence: CALL DETM1 ($n)

n - The statement number to which DETM1 will return if the first three starting points yield a singular matrix.

4.48.8.17 Subroutine Name: DETM2

1. Entry Point: DETM2
2. Purpose: To evaluate the determinant of ND starting points.
3. Calling Sequence: CALL DETM2

4.48.8.18 Subroutine Name: DETM3

1. Entry Point: DETM3
2. Purpose: To iterate for an eigenvalue.
3. Calling Sequence: CALL DETM3 (n_1,n_2,n_3)

n_1 - Return for a new starting point.

n_2 - Return for a new pass through the starting points.

n_3 - Return for problem time expired.

4.48.8.19 Subroutine Name: DETM4

1. Entry Point: DETM4
2. Purpose: To move any starting points necessary.
3. Calling Sequence: CALL DETM4

4.48.8.20 Subroutine Name: DETM5

1. Entry Point: DETM5
2. Purpose: To write out the eigenvalue analysis summary for the determinant method.
3. Calling Sequence: CALL DETM5.

4.48.8.21 Subroutine Name: DETM6

1. Entry Point: DETM6

2. Purpose: To rescale any number by powers of 10.

3. Calling Sequence: CALL DETM6 (D,PØWER)

The arguments are both input and output to the routine and are defined by the following equation.

$$D_{out} \times 10^{PØWER_{out}} = D_{in} \times 10^{PØWER_{in}}, \tag{15}$$

$$.1 \leq |D_{out}| \leq 10 . \tag{16}$$

D is double precision and PØWER is integer.

4.48.8.22 Subroutine Name: FDVECT

1. Entry Point: FDVECT

2. Purpose: To build the load vector to solve for an eigenvector.

3. Calling Sequence: CALL FDVECT (SML1,PK)

SML1 - Smallest diagonal term of [ULL] - real - input.

PK - Accepted eigenvalue - double precision - input.

4.48.8.23 Subroutine Name: EADD

1. Entry Point: EADD

2. Purpose: To compute [A] = [KAA] - p[MAA].

3. Calling Sequence: CALL EADD (P,PREC)

P - Trial value for one eigenvalue - double precision - input.

PREC - "2" - integer - input.

4.48.8.24 Subroutine Name: DETDET

1. Entry Point: DETDET

2. Purpose: To compute the swept determinant of [A].

3. Calling Sequence: CALL DETDET (DETA,IPØWR,P,SML1,ØLDD,IPRØLD)

DETA - Swept determinant of [A] - double precision - output.

IPØWR - Scale factor of DETA - integer - output.

P - Value of p used in computing [A] - double precision - input.

SML1 - Value of smallest diagonal term of [ULL] - real - output.

ØLDD - Swept determinant of previous value of p - double precision - input.

IPRØLD - Scale factor of ØLDD - integer - input.

4.48.8.25 Subroutine Name: ARRM

1. Entry Point: ARRM

2. Purpose: To arrange three starting points in order by the magnitude of their scaled determinants.

3. Calling Sequence: CALL ARRM (P,D,ND)

P - Array of 3 starting points - double precision - input/output.

D - Array of 3 determinants at P - double precision - input/output.

ND - Array of 3 scale factors of D - integer - input/output.

4.48.8.26 Subroutine Name: SUMM

1. Entry Point: SUMM

2. Purpose: To add two scaled numbers together.

3. Calling Sequence: CALL SUMM (SUM,ISUM,T1,IT1,T2,IT2,N)

The arguments are defined by the following equation:

$$SUM \times 10^{ISUM} = T1 \times 10^{IT1} \pm T2 \times 10^{IT2} . \qquad (17)$$

If N = 1, the minus sign is used, otherwise the positive sign is used.

SUM, T1, T2 are double precision.

N, IT1, ISUM, IT2 are integers.

4.48.8.27 Subroutine Name: SQRTM

1. Entry Point: SQRTM

2. Purpose: To compute the square root of a scaled number.

3. Calling Sequence: CALL SQRTM (A,IA,B,IB)

The arguments are defined by the following equation:

$$A \times 10^{IA} = \sqrt{B \times 10^{IB}}$$

A and B are double precision.

IA and IB are integers.

4.48.8.28 Subroutine Name: DETFBS

1. Entry Point: DETFBS

2. Purpose: To solve for the eigenvector.

3. Calling Sequence: CALL DETFBS (F,X,BUFFER,FU,NRØW)

F - Array containing the load vector {F} - double precision - input.

X - Array containing the eigenvector - double precision - output.

BUFFER - GINØ buffer.

FU - Matrix control block for [ULL] - integer - input.

NRØW - Order of the problem (length of F and X) - integer - input.

4.48.8.29 Subroutine Name: VALVEC

1. Entry Point: VALVEC

2. Purpose: To extract the eigenvalues and eigenvectors of a symmetric matrix.

3. Calling Sequence: CALL VALVEC

CØMMØN /GIVN/X1,MØ,MD,MR1,M1,M2,M3,M4,X9(3),RSTRT,X18(82),N,LFREQ,ØRDER,Y4,HFREQ,LAMA,NV,
X8(2),NFØUND,ØEIGS,PHIA,NVER,NEVER,MAX,ITERM

CØMMØN /XXVLVC/Z(1)

/GIVN/ is used by all Givens routines.

MØ — GINØ file number of the input matrix - integer - input.

MD, MR1, M1, M2, M3, M4 - GINØ file numbers of scratch files - integer - input.

RSTRT - '0' indicates no restart is being made - integer - input.

N - Order of the problem - integer - output.

LFREQ, HFREQ - Frequency range for computation of eigenvectors - real - input.

ØRDER - Eigenvalue sort order flag - integer - input.

LAMA, ØEIGS, PHIA - GINØ file name of the associated data blocks - integer - input.

NV - Number of eigenvectors to compute - integer - input.

NFØUND - Number of rigid body modes previously found - integer - input.

NVER - Number of fails to converge on eigenvectors - integer - output.

NEVER - Number of fails to converge on eigenvalues - integer - output.

MAX - Maximum number of QR iterations allowed - integer - input.

ITER - Reason for termination - integer - input.

Z - Open core for VALVEC.

X1, X9, X18, YY, X8 are dummy variables and are not currently used.

4.48.8.30 Subroutine Name: SMLEIG

1. Entry Point: SMLEIG

2. Purpose: To compute the eigenvalues for a 1 by 1 and 2 by 2 matrix and the eigenvector for a 1 by 1 matrix.

3. Calling Sequence: CALL SMLEIG (D,Ø,VAL)

D - Array of diagonal values - double precision - output.

Ø - Array of off-diagonal values - double precision - output.

VAL - Array of eigenvalues - double precision - output.

4.48.8.31 Subroutine TRIDI

1. Entry Point: TRIDI

2. Purpose: To tridiagonalize a symmetric matrix.

3. Calling Sequence: CALL TRIDI (D,Ø,C,A,B,AA)

D - Diagonal terms of the tridiagonal matrix - double precision - output.

Ø - Off diagonal terms of the triadiagonal matrix - double precision - output.

C - Scratch array which contains another copy of the diagonal terms at the conclusion of TRIDI - double precision - output.

A - Remainder of core - double precision - scratch.

B - Scratch array which contains the square of the off-diagonal terms - double precision - output.

AA - Remainder of core - single precision - scratch.

4.48.8.32 Subroutine Name: SICØX

1. Entry Point: SICØX

2. Purpose: To initialize the arrays in SICAS. See section 4.48.8.33.

3. Calling Sequence: CALL SICØX (D,Ø,CØS)

D - Array of diagonal values - double precision - input/output.

Ø - Array of off-diagonal values - double precision - input.

CØS - Array of cosine rotation factors - double precision - input/output.

4.48.8.33 Subroutine Name: SICØX (D,Ø,CØS)

1. Entry Point: SINCAS

2. Purpose: To compute the rotation factors for a given row.

3. Calling Sequence: CALL SINCAS (RØW,FLAG)

RØW - The number of the current row to rotate - integer - input.

FLAG - If no rotations are required for this row, FLAG = 0. Otherwise, FLAG = 1 - integer - input.

4.48.8.34 Subroutine Name: RØTAX

1. Entry Point: RØTAX

2. Purpose: To initialize the arrays in RØTATE. See section 4.48.8.35.

3. Calling Sequence: CALL RØTAX (Ø,SIN,CØS)

Ø - Array of off-diagonal values - double precision - input/output.

SIN - Array of sine rotation factors - double precision - input.

CØS - Array of cosine rotation factors - input.

4.48.8.35 Subroutine Name: RØTAX

1. Entry Point: RØTATE

2. Purpose: To rotate as much of the matrix as fits into core.

3. Calling Sequence: CALL RØTATE (A,RØW,RØW1,RØW2)

A - Partition of the matrix held in core - double precision - input.

RØW -- The row number of current rotation row - integer - input.

RØW1 - The row number of the first row of the matrix partition in core - integer - input.

RØW2 - The row number of the last row of the matrix partition in core - integer - input.

4.48.8.36 Subroutine Name: EMPCØR

1. Entry Point: EMPCØR

2. Purpose: To empty core of a triangular matrix partition.

3. Calling Sequence: CALL EMPCØR (MT1,MT2,PT,PC,FRSRØW,MIDRØW,LASRØW,NX,A,Z)

MT1 - GINØ file name of the first output file - integer - input.

MT2 - GINØ file name of the second output file - integer - input.

PT - Precision (1 = single precision, 2 = double precision) of the input matrix - integer - input.

PC - Precision (1 = single precision, 2 = double precision) of the output matrix - integer - input.

FRSRØW - The row number of the first row in core - integer - input.

MIDRØW - When the current row is greater than MIDRØW, write the remainder of the matrix

onto MT2 - integer - input.

LASRØW - The row number of the last row in core - integer - input.

NX - Order of the matrix - integer - input.

A - Storage containing the triangular matrix - double precision - input.

Z - GINØ buffer - input.

4.48.8.37 Integer Function Name: FILCØR

1. Entry Point: FILCØR

2. Purpose: To fill core with a triangular matrix.

3. Calling Sequence: RØW = FILCØR (MT1,MT2,PC,FRSRØW,MIDRØW,NX,A,NZA,Z)

MT1 - GINØ file name of the first part of the matrix (up to MIDRØW) - integer - input.

MT2 - GINØ file name of the rest of the matrix - integer - input.

PC - Precision (1 = single precision, 2 = double precision) of the matrix in core - integer - input.

FRSRØW - The row number of the first row of the matrix to be read - integer - input.

MIDRØW - Breakpoint of the matrix - integer - input.

NX - Order of the matrix - integer - input.

A - Core storage to hold the matrix - integer - input.

NZA - Number of single precision words at A - integer - input.

Z - GINØ buffer - input.

RØW - The row number of the last row read into A - integer - output.

4.48.8.38 Subroutine Name: QRITER

1. Entry Point: QRITER

2. Purpose: To obtain the eigenvalues of a tridiagonal matrix by the Ortega - Kaiser QR iteration technique.

3. Calling Sequence: CALL QRITER (VAL,Ø,LØC,QR)

VAL - Diagonal terms of the tridiagonal matrix on input, reordered eigenvalues on return - double precision - input/output.

Ø -- Squares of the off-diagonal terms of the tridiagonal matrix. These are iteratively reduced to zero - double precision - input/output.

LØC - Array giving the order in which each eigenvalue is found - integer - output.

QR -- QR ≠ 0 implies that the eigenvalues already exit at VAL and this is an ordering call only - integer - input.

4.48.8.39 Subroutine Name: WILVEC

1. Entry Point: WILVEC

2. Purpose: To compute eigenvectors by the Wilkinson method.

3. Calling Sequence: CALL WILVEC (D,Ø,VAL,VLØC,V,F,P,Q,R,VEC,NX,SVEC)

D - Array of diagonal values - double precision - input.

Ø - Array of off-diagonal values - double precision - input.

VAL - Array of eigenvalues - double precision - input.

VLØC - Array of original ordering of eigenvalues - integer - input.

V,F,P,Q,R - Scratch arrays of length equal to the order of the problem - double precision - input/output.

VEC - Array of core to store vectors - double precision - scratch.

NX - Not used

SVEC - Array of core to store vectors - single precision - scratch.

4.48.8.40 Subroutine Name: INVERT

1. Entry Point: INVERT

2. Purpose: To drive INVTR (see section 4.48.8.41) to compute the inverse of an upper or lower triangular matrix.

3. Calling Sequence: CALL INVERT (IA,IB,SCR1)

CØMMØN /INVTRX/Z(1)

IA - GINØ file name of an upper or lower triangular matrix - integer - input.

IB - GINØ file name of tne inverse of the matrix on IA - integer - input.

SCR1 - GINØ file name of a scratch file - integer - input.

Z - Open core for INVERT.

4.48.8.41 Subroutine Name: INVTR

1. Entry Point: INVTR

2. Purpose: To invert a triangular matrix.

3. Calling Sequence: CALL INVTR (X,DX)

CØMMØN /INVTRX/FA(7),FB(7),SCRFIL,NX,PREC

X - Open core for INVTR - single precision - scratch.

DX - Open core for INVTR (same address as X) - double precision - scratch.

FA - Matrix control block for the input matrix - integer - input.

FB - Matrix control block for the output matrix - integer - input.

SCRFIL - GINØ file name of a scratch file - integer - input.

NX - Length in words of X - integer - input.

PREC - Precision (1 = single precision, 2 = double precision) of the computation - integer - input.

4.48.8.42 Subroutine Name: READ6

1. Entry Point: READ6

2. Purpose: To merge the rigid body eigenvectors with vectors computed by GIVENS.

3. Calling Sequence: CALL READ6 (IRBM,IMGIV,NR,IPHIA)

CØMMØN /READ6X/Z(1)

IRBM - GINØ file name for eigenvectors computed by READ1 - integer - input.

IMGIV - GINØ file name for eigenvectors computed by GIVENS - integer - input.

NR - Number of eigenvectors computed by READ1 - integer - input.

IPHIA - GINØ file name for merged eigenvectors - integer - input.

Z - Open core for READ6.

4.48.9 Design Requirements

Design requirements are peculiar to the method chosen. The appropriate subroutines should be consulted.

4.48.10 Diagnostic Messages

Messages output from READ are peculiar to the individual subroutine.

4.48.11 Mathematical Considerations for the Inverse Power Method

The algorithm for finding the eigenvalues and eigenvectors of

$$([K] - \lambda[M])\,[\Phi] = 0, \tag{19}$$

is given as follows.

1. The iteration algorithm is given by the following equations.

$$([K] - \lambda_o\,[M])\,\{W_n\} = [M]\,\{u_{n-1}\}\,, \tag{20}$$

$$\{\bar{u}_n\} = \frac{1}{C_n}\,\{W_n\}, \tag{21}$$

$$\{u_n\} = \{\bar{u}_n\} - \sum_i \left(\{\Phi_i\}^T\,[M]\,\{\bar{u}_n\}\right)\,\{\Phi_i\}\,, \tag{22}$$

where C_n is the absolute value of the maximum component of $\{W_n\}$. The sum over i extends over all previously extracted eigenvectors.

2. Form

$$\{F_n\} = [M]\,\{u_n\}. \tag{23}$$

3. Compute

$$\alpha_n = \left(\{u_n\}^T\,\{F_n\}\right)^{1/2}. \tag{24}$$

4. Compute

$$\{\delta u_n\} = \frac{\{u_n\}}{\alpha_n} - \frac{\{u_{n-1}\}}{\alpha_{n-1}}. \tag{25}$$

5. Compute

$$\{\delta F_n\} = [M]\,\{\delta u_n\}. \tag{26}$$

6. Compute the approximate eigenvalue

$$\lambda_1 = \frac{1}{C_n}\,\frac{\alpha_{n-1}}{\alpha_n}. \tag{27}$$

7. For the rapid convergence test, compute

$$\eta = (\{\delta u_n\}^T \{\delta F_n\})^{1/2}. \tag{28}$$

8. For normal convergence, compute

$$\frac{\lambda_2}{\lambda_1} = \frac{\{\delta u_n\}^T \{\delta F_{n-1}\}}{\eta^2}, \tag{29}$$

and

$$\delta_n = (\{\delta u_n\}^T \{\delta F_n\}) / (1 - \frac{\lambda_2}{\lambda_1})^2. \tag{30}$$

9. For the shift decision test, compute

$$h_{1,n} = \frac{\lambda_{1,n} - \lambda_{1,n-1}}{R_o}, \tag{31}$$

and

$$k = \frac{\log_{10}\left(\frac{\sqrt{\delta_n}}{A\varepsilon}\right)}{\log_{10}(\lambda_2/\lambda_1)}. \tag{32}$$

10. For the λ_2 reliability test, compute

$$h_{2,n} = \frac{\lambda_{2,n} - \lambda_{2,n-1}}{R_o} \tag{33}$$

The above equations, along with the proper logic given in Figure 1 form the basis for the Inverse Power Method.

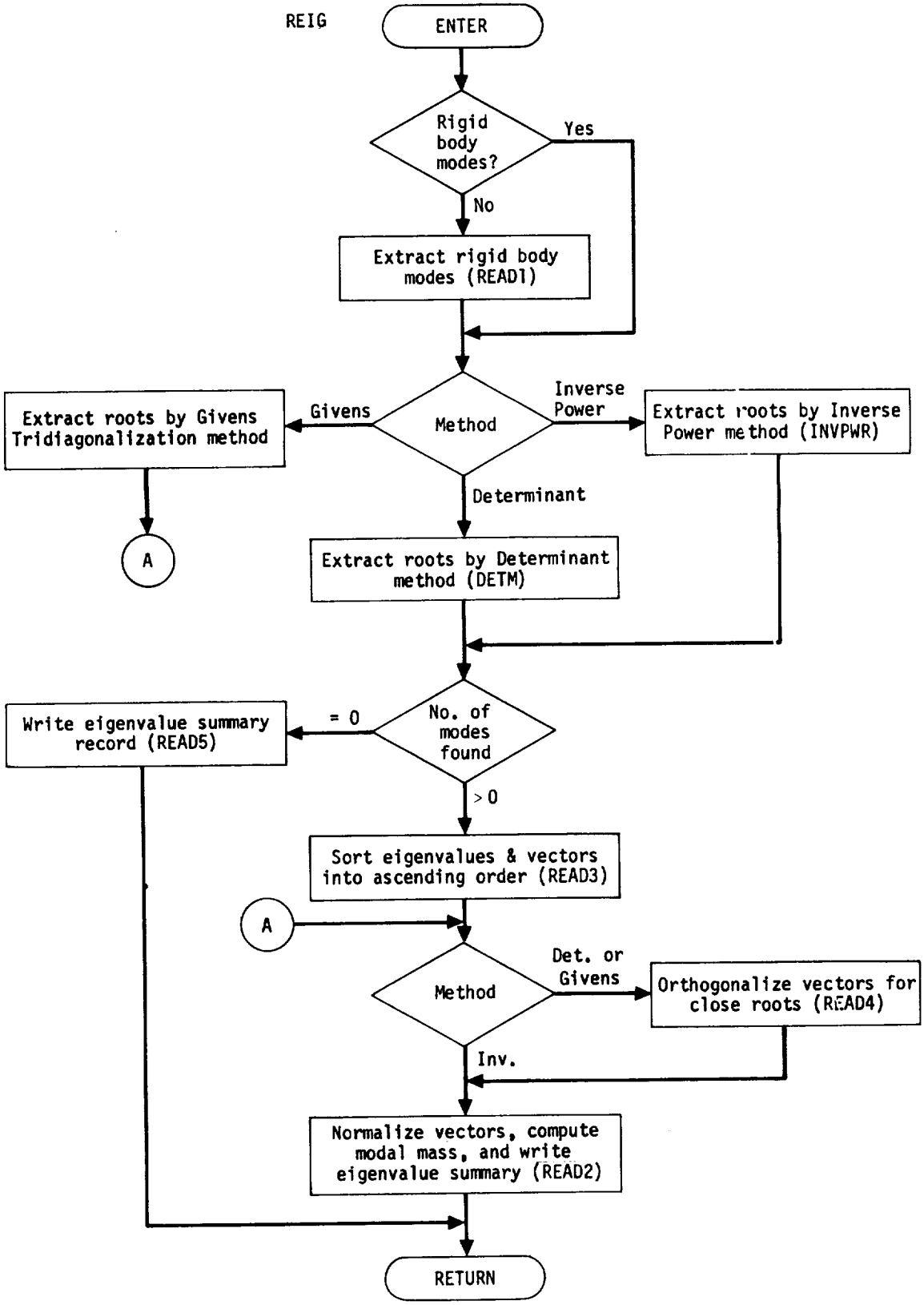


Figure 1.(a) Flowchart for module READ.

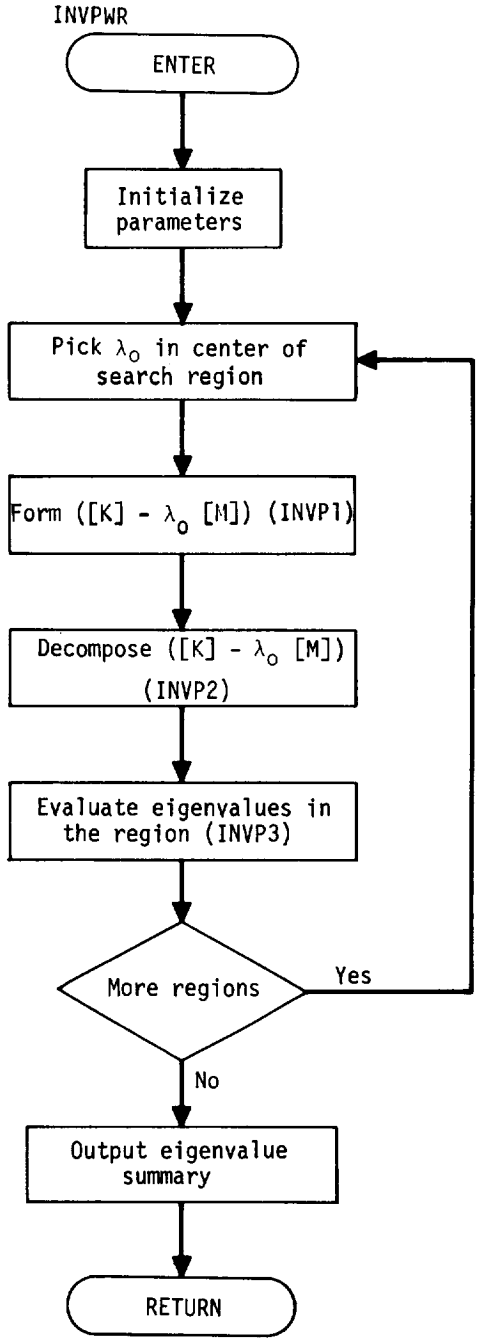


Figure 1.(b) Flowchart for module READ

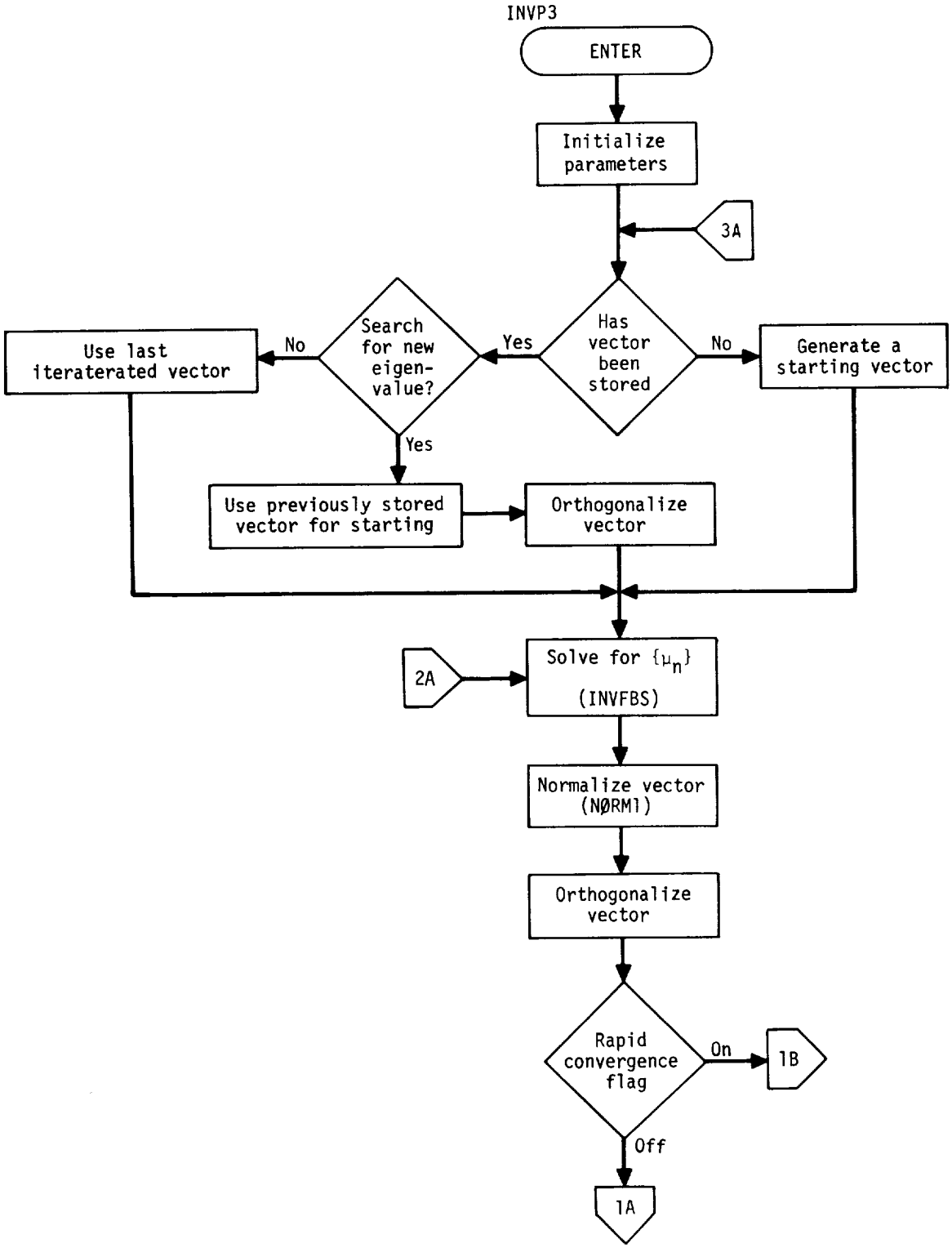


Figure 1.(c) Flowchart for module READ

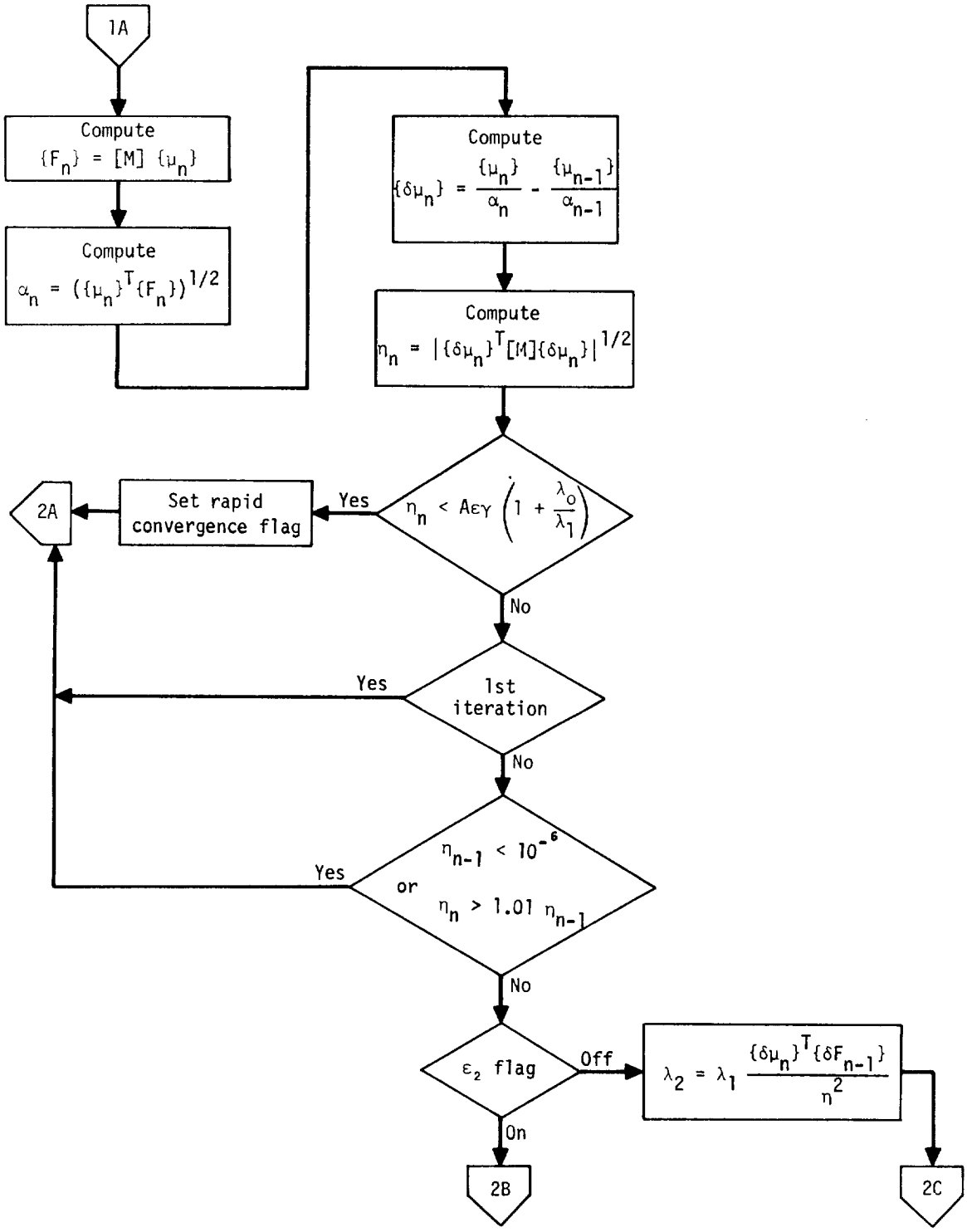


Figure 1.(d) Flowchart for module READ

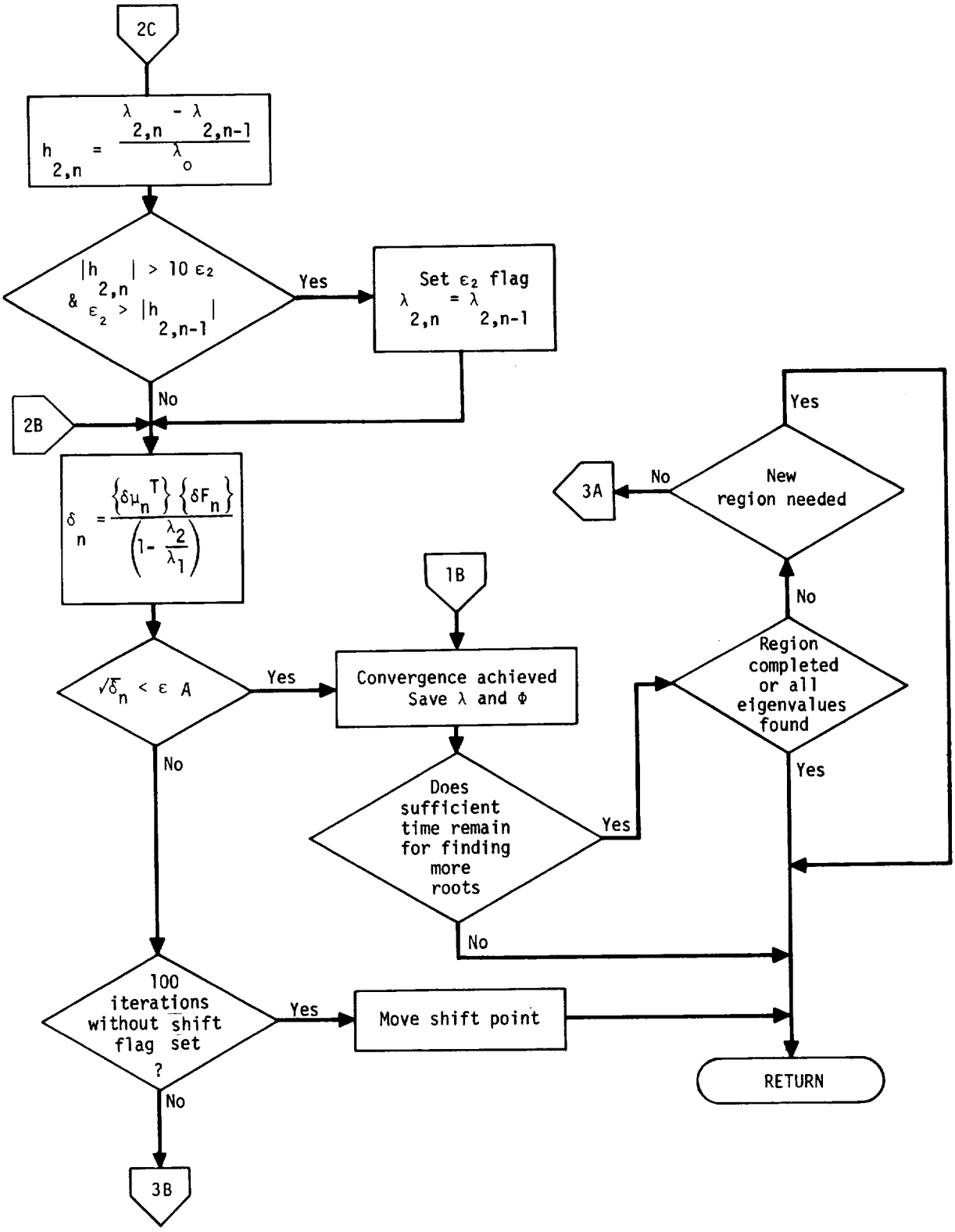


Figure 1.(e) Flowchart for module READ.

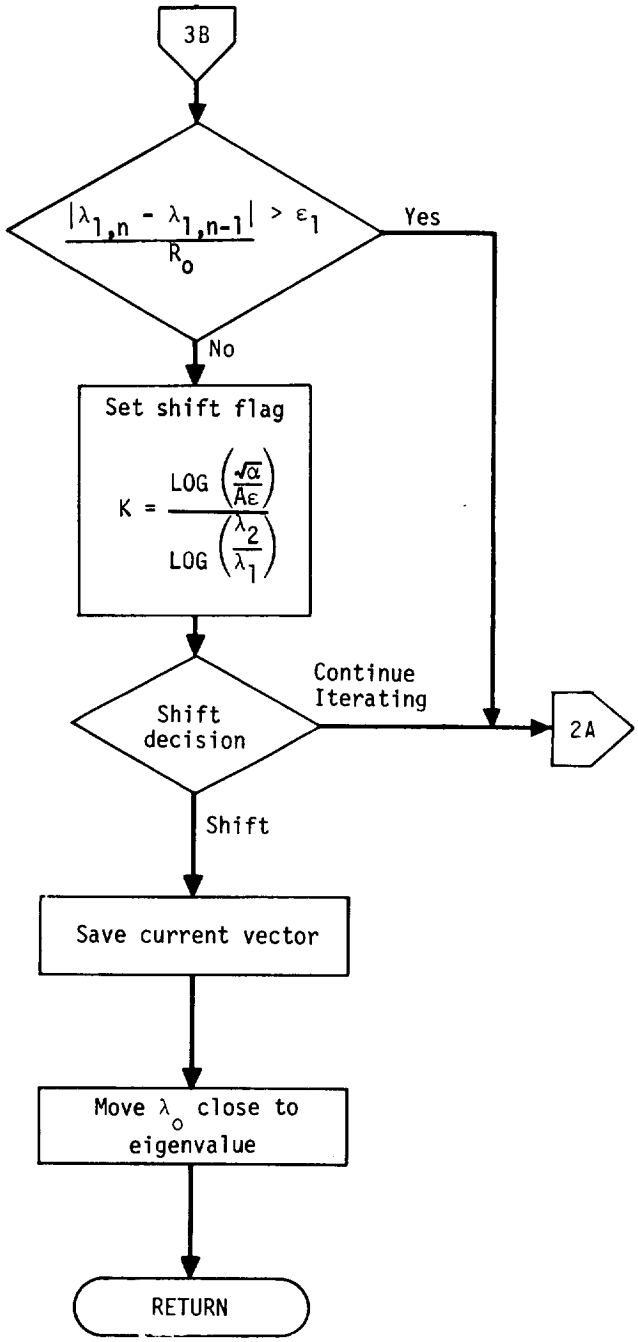


Figure 1.(f) Flowchart for module READ

4.49 FUNCTIONAL MODULE DSMG1 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR - PHASE 1)

4.49.1 Entry Point: DSMG1

4.49.2 Purpose

To generate the differential stiffness matrix, $[K^{d}_{gg}]$, which is used in two Rigid Formats: Static Analysis with Differential Stiffness and Buckling Analysis.

4.49.3 DMAP Calling Sequence

```
DSMG1     CASECC,GPTT,SIL,EDT,UGV,CSTM,MPT,ECPT,GPCT,DIT/KDGG/V,N,DSCØSET    $
```

4.49.4 Input Data Blocks

CASECC - Case Control Data Table.

GPTT - Grid Point Temperature Table.

SIL - Scalar Index List.

EDT - Element Deformation Table.

UGV - Initial Approximation to the Displacement Vector - g set.

CSTM - Coordinate System Transformation Matrices.

MPT - Material Properties Table.

ECPT - Element Connection and Properties Table.

GPCT - Grid Point Connection Table.

DIT - Direct Input Tables.

Notes:

1. A fatal error exists if CASECC is purged.
2. A fatal error exists if there is a temperature load and the GPTT is purged.
3. A fatal error exists if there is a temperature load and the SIL is purged.
4. A fatal error exists if an element deformation set is requested by the user in case control and EDT is purged.
5. A fatal error exists if UGV is purged.
6. CSTM can be purged. However, if some grid point of the model is not in basic coordinates and the CSTM is purged, a fatal error occurs.

7. If the MPT is purged and some element references a material property, a fatal error occurs.

8. A fatal error occurs if the ECPT is purged.

9. A fatal error occurs if the GPCT is purged.

10. If some material property is temperature dependent, DIT cannot be purged.

4.49.5 Output Data Blocks

KDGG - Partition of differential stiffness matrix - g set.

Note: KDGG cannot be pre-purged. A fatal error occurs if it is.

4.49.6 Parameters

DSCØSET - Output-integer-no default value. The set identification number of a DSFACT bulk data card chosen by the user in his Case Control Deck. If no such set was specified by the user, DSCØSET is set to -1.

4.49.7 Method

The module driver, DSMG1, is a very short routine whose only function is to call the two principal subroutines of the module, DS1 and DS1A, which accomplish the two phases of the module. The first phase of the module is incorporated in subroutine DS1. This routine creates the scratch file ECPTDS (GINØ file number 301) by appending to each element in the ECPT data block for which differential stiffness is defined an element deformation, an average element loading temperature for all elements except the conical shell element (a loading temperature at each grid point is appended for the conical shell element) and the proper components of the displacement vector, UGV. It should be noted that although element deformations are defined only for rods, tubes, beams, and bars, an element deformation is attached to each element written on the ECPTDS scratch data block. The elements admissible to the ECPTDS scratch data block are: rods, beams, tubes, shear (but not twist) panels, triangular and quadrilateral elements (TRMEM's and QDMEM's), the combination membrane and plate triangular and quadrilateral elements (TRIA1, TRIA2, QUAD1, QUAD2) and the conical shell element.

The flow of DS1 is given in the following steps:

1. The length of variable core is determined, buffers are defined at the bottom of open

core, and zero pointers to, and lengths of, subarrays in open core are initialized.

2. The Case Control data block, CASECC, is read to determine the element deformation set number, the loading temperature set number, and the differential stiffness coefficient set number.

3. If there is no temperature load, go to step 4. If there is a temperature load, the Grid Point Temperature Table data block, GPTT, is read such that the proper temperature set is read into open core. The Scalar Index List data block, SIL, is read into open core, and the internal grid point numbers in the GPTT are replaced with the corresponding SIL numbers.

4. If a non-zero element deformation set number was read from CASECC, the element deformation set is read into open core.

5. The first (and only) record of the displacement vector file, UGV, is read into open core, and then the ECPTDS and ECPT files are opened and positioned correctly.

6. The ECPT data block is read record by record, and the ECPTDS scratch data block is generated for use by subroutine DS1A. The procedure for each record is as follows. The pivot point is read. For each element in the current ECPT record, it is determined if the element type is admissible to the ECPTDS data block. If is not admissible, the next element entry is read; if it is admissible, a test is made to determine if the element type is a TRIA1, QUAD1 or conical shell element. If it is a TRIA1, QUAD1 or a conical shell element and if this element has zero membrane thickness (which implies the element has bending properties only) and since differential stiffness is not defined for plate elements, then the next element entry is read from the current ECPT record. When the first element of the ECPT record which belongs to the differential stiffness set is encountered, the pivot point is written on the ECPTDS data block. Note that if the element is a BAR, the ECPT entry is rearranged to conform with the ECPT entry for the BEAM so that the DBEAM subroutine may be called in subroutine DS1A. The element type is written on the ECPTDS record. Then an element deformation number, an average element loading temperature (two loading temperatures for a conical shell element), and the displacement vector components are appended in core to the ECPT entry for the element. Finally this appended ECPT entry is written on ECPTDS. It should be noted that the 2nd through the $(n+1)^{st}$ words (n being the number of grid points of the element) of every ECPT entry, since they are scalar index numbers, are direct pointers

into the displacement vector, UGV. Note further that for elements for which differential stiffness is defined (save for the BAR, the BEAM and the conical shell element) only the three translational components of the displacement vector at each grid point are appended. For the BAR and BEAM all six displacement components at each grid point are appended. For the special case of the conical shell element, only the six displacement components at each grid point associated with the zeroth harmonic are appended.

When an end-of-record is sensed for an ECPT record, an indicator is interrogated to determine whether or not any of the elements in the record were written on the ECPTDS data block. If there weren't any, a -1 is written on ECPTDS with an end-of-record mark; if there was at least one element, an end-of-record is written on ECPTDS. In either case, after this step has been done, the next ECPT record is processed identically. When an end-of-file is sensed on the ECPT, the files for ECPT and ECPTDS are closed, and the routine returns to DSMG1.

The module driver, DSMG1, tests the argument of DS1. If it is equal to zero, this implies that no element of the ECPT was a member of the set of elements for which differential stiffness is defined, and hence a fatal error exit occurs. If the argument is greater than zero, subroutine DS1A is called.

In the second phase of the module, subroutine DS1A processes the scratch file ECPTDS to produce the differential stiffness matrix, $[K^d_{gg}]$. The logic of this processing is very similar to that used in subroutine SMA1A (of module SMA1, the stiffness matrix generation routine). See the Module Functional Description for SMA1 (section 4.27) for details.

4.49.8 Subroutines

All subroutines defined below except DS1 are necessary to accomplish the second phase of the module, which is the processing of the ECPTDS scratch file in order to generate $[K^d_{gg}]$. The auxiliary routines PRETRD, PREMAT, GMMATD and INVERD are used similarly to module SMA1.

The data (an ECPTDS entry) input to an element differential stiffness matrix generation routine (e.g., DRØD) are communicated to the routine via /DS1AET/, which fact is not expressly stated in the subroutine descriptions given below.

4.49.8.1 Subroutine Name: DS1

1. Entry Point: DS1

2. Purpose: See discussion above.

3. Calling Sequence: CALL DS1 (IARG)

IARG - Initially set to zero in subroutine DS1. This variable is set to 1 in DS1 every time an element in the ECPT is encountered which is in the set of elements for which differential stiffness is defined. If IARG is still zero upon DS1's return to DSMG1, DSMG1 will terminate the job by calling MESAGE and PEXIT.

4.49.8.2 Subroutine Name: DS1A

1. Entry Point: DS1A

2. Purpose: See discussion above.

3. Calling Sequence: CALL DS1A

4.49.8.3 Subroutine Name: DS1B

1. Entry Point: DS1B

2. Purpose: This routine, called by the module's element matrix generation routines such as DRØD, DBEAM, etc., adds a double precision 6 by 6 element differential stiffness matrix to the "submatrix" corresponding to the current pivot point. This same function is performed in module SMA1 by subroutine SMA1B, in SMA2 by SMA2B and in PLA4 by PLA4B.

3. Calling Sequence: CALL DS1B (KE,J)

KE - Row-stored double precision 6 by 6 matrix to be added to the "submatrix" corresponding to the current pivot point. Double precision; input.

J - The column index of the $[K^{d}_{gg}]$ matrix which corresponds to the first column of the KE matrix. Integer; input.

4.49.8.4 Block Data Subprogram Name: DS1ABD

1. Entry Point: DS1ABD

2. Purpose: This block data program sets 1) GINØ file numbers; 2) GINØ ØPEN, WRITE and CLØSE option parameters; 3) differential stiffness element routine (e.g., DRØD,DBEAM) overlay parameters; and 4) an array which defines the number of words to be read for each element from the ECPTDS scratch file. The common block /DS1AAA/ is used for the second phase of the module.

4.49.8.5 Subroutine Name: DRØD

1. Entry Point: DRØD

2. Purpose: To generate the element differential stiffness matrix for a RØD, CØNRØD or TUBE element.

3. Calling Sequence: CALL DRØD

4.49.8.6 Subroutine Name: DBEAM

1. Entry Point: DBEAM

2. Purpose: To generate the element differential stiffness matrix for a BAR or BEAM element.

3. Calling Sequence: CALL DBEAM

4.49.8.7 Subroutine Name: DSHEAR

1. Entry Point: DSHEAR

2. Purpose: To generate the element differential stiffness matrix for a SHEAR (panel) element.

3. Calling Sequence: CALL DSHEAR

4.49.8.8 Subroutine Name: DTRMEM

1. Entry Point: DTRMEM

2. Purpose: To generate the element differential stiffness matrix for a TRIA1, TRIA2 and TRMEM element.

3. Calling Sequence: CALL DTRMEM (IARG)

$$\text{IARG} = \begin{cases} 0 = \text{Called from DSGM1} \\ 1 = \text{Called from DQDMEM} \end{cases}$$

Note that the ECPTDS entry for the TRIA1 element is rearranged in subroutine DS1A to conform with the ECPTDS entry for the TRMEM element before DTRMEM is called.

4.49.8.9 Subroutine Name: DQDMEM

1. Entry Point: DQDMEM

2. Purpose: To generate the element differential stiffness matrix for a QUAD1, QUAD2 and QDMEM element.

3. Calling Sequence: CALL DQDMEM

Note that the ECPTDS entry for the QUAD1 element is rearranged in subroutine DS1A to conform with the ECPTDS entry for the QDMEM element before DQDMEM is called.

4.49.8.10 Subroutine Name: DCØNE

1. Entry Point: DCØNE

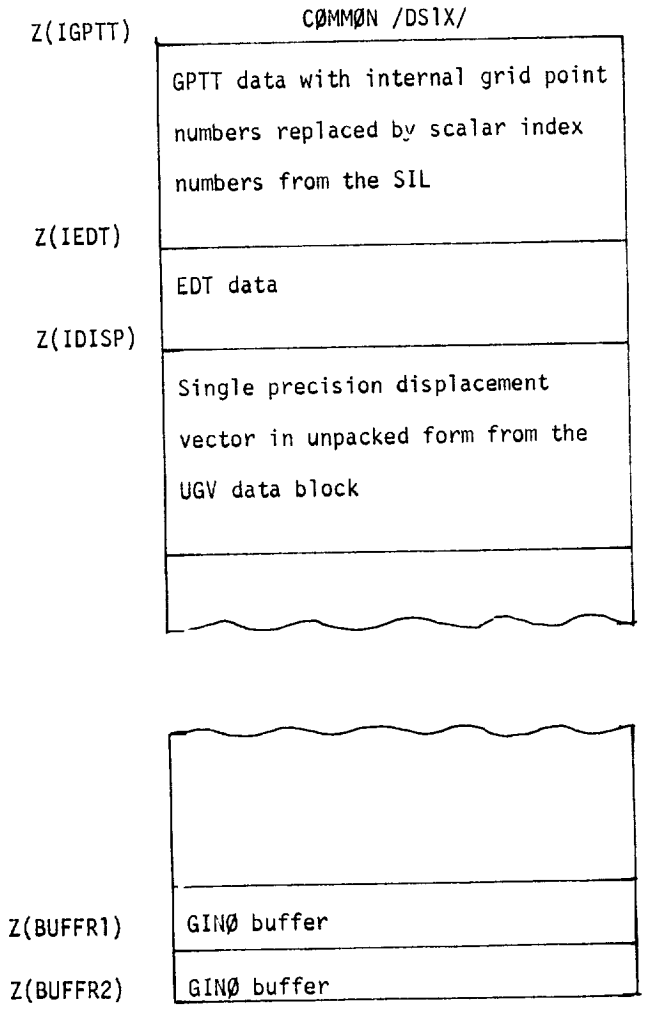
2. Purpose: To generate the element differential stiffness matrix for a conical shell element.

3. Calling Sequence: CALL DCØNE

4.49.9 Design Requirements

4.49.9.1 Open Core Design

During phase one of the module's operation, the maximum core storage requirements are given in the following diagram, in which open core, the Z array, is defined at /DS1X/.



During phase two of the module's operation, the open core design is the same as that for module SMA1, except that only 3 GINØ buffers are needed instead of 5, one for ECPTDS, one for GPCT and one for KDGG. Open core for phase two is defined at /DS1AXX/.

4.49.9.2 Common Storage Requirements

During phase one, there are no special common storage requirements.

During phase two, /DS1AAA/ takes the place of /SMA1IØ/, /SMA1BK/ and /SMA1CL/; /DS1AET/ corresponds to /SMA1ET/; and /DS1ADP/ corresponds to /SMA1DP/. See the common storage requirements section of the SMA1 Module Functional Description for details on /SMA1IØ/, /SMA1BK/, /SMA1CL/, /SMA1ET/ and /SMA1DP/ (section 4.27.9.3).

4.49.9.3 Arithmetic Considerations

All floating point arithmetic operations are carried out in double precision. $[K^{c}_{gg}]$ is a real double precision symmetric matrix.

4.49.10 Diagnostic Messages

During phase one the fatal messages 2029 and 2030 can be called if the GPTT is not in the proper format. If a BAR element is encountered whose coupled bending interia property, I_{12}, is non-zero, then it is set to zero and the user is warned of this fact with message 2111. User messages 2081 and 2083 are fatal messages indicating a null differential stiffness matrix and a null displacement vector respectively.

For phase two diagnostics, the reader is referred to the diagnostic message section of the Module Functional Description for SMA1.

4.50 FUNCTIONAL MODULE SMP2 (STRUCTURAL MATRIX PARTITIONER - PHASE 2)

4.50.1 Entry Point: SMP2

4.50.2 Purpose

To perform the following matrix operations:

$$[K^d_{ff}] \Longrightarrow \left[\begin{array}{c|c} \bar{K}^d_{aa} & K^d_{ao} \\ \hline (K^d_{ao})^T & K^d_{oo} \end{array}\right], \tag{1}$$

$$[K^d_{aa}] = [\bar{K}^d_{aa}] + [K^d_{ao}]\,[G_o] + ([K^d_{ao}]\,[G_o])^T + [G_o]^T\,[K^d_{oo}]\,[G_o]. \tag{2}$$

4.50.3 DMAP Calling Sequence

```
SMP2    USET,GØ,KDFF/KDAA/  $
```

4.50.4 Input Data Blocks

USET	-	Displacement set definitions table.
GØ	-	Structural matrix partitioning transformation matrix.
KDFF	-	Partition of differential stiffness matrix - f set.

4.50.5 Output Data Blocks

KDAA	-	Partition of differential stiffness matrix - a set.

4.50.6 Parameters

None

4.50.7 Method

The matrix $[K^d_{ff}]$ is partitioned as in Equation 1 using USET (UF,UA,UØ), and matrix subroutine ELIM (see section 3.5.22 for details) is called to perform the operations in Equation 2.

4.50.8 Subroutines

SMP2 has no auxiliary subroutine.

4.50.9 Design Requirements

See design requirements for subroutine UPART and its entry point MPART which perform the symmetric partition of Equation 1; also see design requirements for subroutine ELIM. These are in sections 3.5.9 and 3.5.22 respectively.

4.51 FUNCTIONAL MODULE DSMG2 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR - PHASE 2).

4.51.1 Entry Point: DSMG2

4.51.2 Purpose

This module performs the following matrix operations:

$$[K^b_{\ell\ell}] = [K_{aa}] + \beta[K^d_{aa}] , \tag{1}$$

$$[K^b_{fs}] = [K_{fs}] + \beta[K^d_{fs}], \tag{2}$$

$$[K^b_{ss}] = [K_{ss}] + \beta[K^d_{ss}] , \tag{3}$$

$$\{P^b_{\ell}\} = \beta\{P_{\ell}\} , \tag{4}$$

$$\{P^b_{s}\} = \beta\{P_{s}\} , \tag{5}$$

$$\{Y^b_{s}\} = \beta\{Y_{s}\} , \tag{6}$$

$$\{u^{ob}_{o}\} = \beta\{u^{o}_{o}\} . \tag{7}$$

The value of β is on a DSFACT bulk data card whose set identification number is specified by the input parameter DSCØSET. The particular value on that card to be used for β on any pass through the DMAP loop in the Static Analysis with Differential Stiffness Rigid Format is controlled by the parameter NDSKIP (see below).

4.51.3 DMAP Calling Sequence

```
DSMG2    MPT,KAA,KDAA,KFS,KDFS,KSS,KDSS,PL,PS,YS,UØØV/KBLL,KBFS,KBSS,PBL,PBS,YBS,UBØØV/
         V,N,NDSKIP/V,N,REPEATD/V,N,DSCØSET  $
```

4.51.4 Input Data Blocks

MPT - Material Property Table.

KAA - Partition of stiffness matrix - a set.

KDAA - Partition of differential stiffness matrix - a set.

KFS - Partition of stiffness matrix after single-point constraints have been removed.

KDFS - Partition of differential stiffness matrix after single-point constraints have been removed.

KSS - Partition of stiffness matrix after single-point constraints have been removed - s set.

KDSS - Partition of differential stiffness matrix after single-point constraints have been removed - s set.

PL - Partition of the load vector matrix giving static loads on ℓ set.

PS - Partition of the load vector matrix giving loads in s set.

YS - Constrained displacement vector - s set.

UØØV - Partition of the displacement vector matrix giving displacements in the o set.

Notes:

1. A fatal error occurs if MPT is purged.
2. KAA and KDAA cannot be purged.
3. KFS and KDFS must be both purged or both non-purged.
4. KSS and KDSS must be both purged or both non-purged.
5. A fatal error occurs if PL is purged.
6. PS, YS and UØØV can be purged.

4.51.5 Output Data Blocks

KBLL - Partition of the stiffness matrix of the first order approximation to large displacements - ℓ set.

KBFS - Partition of the stiffness matrix of the first order approximation to large displacements.

KBSS - Partition of the stiffness matrix of the first order approximation to large displacements - s set.

PBL - Partition of the load vector of the first order approximation to the large displacements - ℓ set.

PBS - Partition of the load vector of the first order approximation to the large displacements - ℓ set.

YBS - Partition of the constrained displacement vector of the first order approximation to the large displacement vector - s set.

UBØØV - Partition of the displacement vector of the first order approximation to the large displacement problem - o set.

Notes:

1. KBLL must not be purged or a fatal error exists.
2. KBFS can be purged if and only if both KFS and KDFS are purged.
3. KBSS can be purged if and only if both KSS and KDSS are purged.
4. PL must not be purged or a fatal error exists.
5. PBS, YBS and UBØØV can be purged if and only if PS, YS and UBØØV respectively are purged.

4.51.6 Parameters

NDSKIP - Input and output-integer - must be set to zero before the DMAP loop for Static Analysis with Differential Stiffness is initiated. This parameter is used as a loop counter for the DMAP loop in the Static Analysis with Differential Stiffness Rigid Format. It enables this module to skip the proper number of words on the proper DSFACT card (see definition of the parameter DSCØSET below) to fetch the value of β for the scalar multiplications defined above.

REPEATD - Output-integer-no default value. This parameter is set to +1 if another pass is to be made through the Static Analysis with Differential Stiffness DMAP loop (and hence through the DSMG2 module). It is set to -1 if the module determines that the current pass through the module will be the final one. This latter condition is met if (1) no differential stiffness coefficient set number was specified by the user in his Case Control Deck (if this condition is true, module DSMG1 sets the parameter DSCØSET (see below) equal to -1 and a value of 1.0 is used for β); or if (2) the current β is the last one on the user specified DSFACT card.

DSCØSET - Input-integer-no default value. DSCØSET is the differential stiffness coefficient

set number of a DSFACT bulk data card chosen by the user in his Case Control Deck. If no such set was specified by the user, DSCØSET = -1 and this module will set $\beta = 1.0$ and set the REPEATD parameter to -1. If DSFACT > 0, DSFACT is used to search the MPT for the DSFACT bulk data card chosen by the user.

4.51.7 Method

REPEATD is set to one and NDSKIP is incremented by one. If DSCØSET = -1, REPEATD is set to -1 and the three matrix additions in Equations 1, 2 and 3 are carried out using the matrix subroutine SSG2C. It should be noted that if DSCØSET = -1, it is assumed that DMAP equivalences have been made between PL and PBL, PS and PBS, YS and YBS, and UØØV and UBØØV and hence Equations 4 through 7 are not calculated.

If DSCØSET $\neq$ -1, the MPT is searched until a match is found between DSCØSET and a DSFACT bulk data card image on the MPT. If a match is found, the parameter NDSKIP is used to find the correct value of β for this pass through the DMAP loop. If a match is not found, a fatal error occurs. A match having been found, the following operations are performed.

1. REPEATD is set to -1 if it is determined that this β is the last one on the DSFACT card.
2. The four scalar multiplications in Equations 4 through 7 are performed using matrix subroutine SSG2C.
3. The three matrix additions in Equations 1 through 3 are performed using matrix subroutine SSG2C.

4.51.8 Subroutines

DSMG2 has no auxiliary subroutines. A description of SSG2C can be found in section 3.5.11.

4.51.9 Design Requirements

If DSCØSET = -1, it is assumed that DMAP equivalences have been made between the data blocks corresponding to the matrices in Equations 4 through 7.

Two GINØ buffers are needed and open core is defined at /DSMG2X/.

4.51.10 Diagnostic Messages

Fatal error message 2084 may occur.

4.52 FUNCTIONAL MODULE PLA1 (PIECEWISE LINEAR ANALYSIS - PHASE 1)

4.52.1 Entry Point: PLA1

4.52.2 Purpose

PLA1 is a pre-processor for the modules unique to the Piecewise Linear Analysis Rigid Format. PLA1 extracts the linear elements from the ECPT data block (an element is defined to be linear if its modulus of elasticity - E on a MAT1 bulk data card - is not defined to be the first derivative of a stress-strain tabular function defined on a TABLES1 bulk data card) to form $[K_{gg}^{x\ell}]$, extracts the nonlinear element entries from the ECPT data block to form ECPTNL, and separates the linear and nonlinear element entries in the EST to form ESTL and ESTNL.

4.52.3 DMAP Calling Sequence

```
PLA1    CSTM,MPT,ECPT,GPCT,DIT,CASECC,EST/KGGXL,ECPTNL,ESTL,ESTNL/V,N,KGGLPG/V,N,NPLALIM/
        V,N,ECPTNLPG/V,N,PLSETNØ/V,N,NØNLSTR/V,N,PLFACT    $
```

4.52.4 Input Data Blocks

CSTM	-	Coordinate System Transformation Matrices.
MPT	-	Material Properties Table.
ECPT	-	Element Connection and Properties Table.
GPCT	-	Grid Point Connection Table.
DIT	-	Direct Input Tables.
CASECC	-	Case Control Data Table.
EST	-	Element Summary Table.

Notes: 1. The CSTM may be purged. However, if it is, and some grid point is not in basic coordinates, then a fatal error exists.

2. If an element references a material property and the MPT is purged, a fatal error exists.

3. If any of the data blocks ECPT, GPCT, DIT, CASECC or EST is purged a fatal error exists.

Rigid Format DMAP loop counter. It is used in this routine to find the proper loading factors on the PLFACT bulk data card specified by the user (see PLSETNØ below).

PLTSETNØ - Input-integer-no default value. PLSETNØ is the set identification number of some PLFACT bulk data card chosen by the user in his Case Control Deck. It is used to find this PLFACT card in the MPT data block.

4.54.7 Method

The module driver, PLA3, is a short routine whose only function is to call subroutines PLA31 and PLA32 which accomplish phase 1 and phase 2 of the task of the module respectively. Subroutine PLA31 reads the incremental displacement vector into core and appends to each element entry of the ESTNL data block the components of the incremental displacement vector corresponding to the grid points of each element. This merged information is written on the scratch data block ESTNLS, GINØ file number 301. In PLA32, the ESTNLS data block is read, and the proper element routine is called to compute element stresses which are prepared in ØFP (Output File Processor) format. Each element routine also updates incremental stress data. The ESTNL data for each element with the updated stress information (but without the components of the displacement vector) are written on ESTNL1.

Ir PLA31, for TRMEM and QDMEM elements, only the three translational components of the displacement vector at each grid point of the element are appended to the ESTNL entry. Other elements for which Piecewise Linear Analysis is defined use all six components at each grid point.

In PLA32, the difference quotients γ^* and γ, which are the previous and current (with respect to the DMAP loop in the PLA Rigid Format) load increment ratios, are computed as follows. Let $P_1, P_2, P_3, \ldots,$ be the loading factors on a PLFACT bulk data card. Define $P_o = 0$. Define

$$\alpha_i = P_i - P_{i-1} \quad , \tag{1}$$

for $i \geq 1$. Then, define $\gamma_1^* = 0$, and

$$\gamma_i^* = \frac{\alpha_i}{\alpha_{i-1}} \quad , \tag{2}$$

element stress output for a set of elements all of whose members are linear, then NØNLSTR is set to -1. If there is a stress output request for some nonlinear element, then NØNLSTR is set = +1 and PLA3 will be called each time through the PLA DMAP loop.

PLFACT - Output-complex-no default value. The first load increment factor to be used the first time through the PLA DMAP loop.

4.52.7 Method

The routine is divided into two phases. Phase 1 processes the ECPT data block in a fashion similar to module SMA1 (see section 4.27). For each pivot point, every element is examined to determine whether it is linear or non linear. This is accomplished by calling subroutine MAT (see section 3.4.36) with the second word, INFLAG, of the common block /MATIN/ equal to 5. If the element is linear, the proper element stiffness matrix generator routine such as KRØD, KBAR, etc. is called. The element routine will, in turn, call subroutine SMA1B to add its contribution to the 6 (or fewer, if the pivot point is a scalar point or there is not enough core storage available-see Module Functional Description for SMA1, section 4.27-) rows of the KGGXL matrix currently being generated. If the MAT routine determines that the element is nonlinear, then the ECPT entry for that element, along with words needed subsequently in module PLA4, is appended (see data block description for ECPTNL in section 2.3.34.4 for details). The number of words appended depends upon element type. This appended ECPT entry is written onto the ECPTNL data block.

Phase 2 of this routine reads the first record of CASECC into core, and the MPT data block is searched to find the set number on a PLFACT bulk data card requested by the user in CASECC. Parameters PLFACT and PLASETNØ are set. The EST data block is then processed. The logic here is similar to Phase 1. For each element, it is determined if the element is linear or nonlinear. If the element is linear, its EST entry is copied onto the ESTL data block. If the element is non-linear and stress output is requested, the EST entry along with words needed subsequently in module PLA3 are appended. The number of words appended depends upon element type and is not the same number of words appended to the ECPT entry to create the ECPTNL. If the element is nonlinear and stress output is not requested, the EST entry for the element is written onto the ESTL data block.

4.52.8 Subroutines

PLA1 has no auxiliary subroutines as such. However, it uses all the structural element routines of module SMA1 to generate $\left[K_{gg}^{Xf}\right]$, which in turn use the common blocks of SMA1 as well as the "insertion" routine, SMA1B. See Module Functional Description for SMA1, section 4.27.

4.52.9 Design Requirements

For phase 1 of PLA1, the design requirements are the same as those for SMA1. For phase 2, the first record of CASECC must be able to be held in open core.

4.52.10 Diagnostic Messages

For phase 1, see diagnostic messages for module SMA1, section 4.27.10. For phase 2, a user fatal message, 2032, occurs if the PLFACT bulk data card which was chosen by the user in his Case Control Deck could not be found in the MPT.

4.53 FUNCTIONAL MODULE PLA2 (PIECEWISE LINEAR ANALYSIS - PHASE 2)

4.53.1 Entry Point: PLA2

4.53.2 Purpose

To add the incremental displacement vector, the incremental load vector, and the incremental vector of single-point forces of constraint for the current pass through the Piecewise Linear Analysis Rigid Format DMAP loop to the current running sum of these vectors:

$$\{u_{g_{i+1}}\} = \{u_{g_i}\} + \{\Delta u_{g_i}\} , \tag{1}$$

$$\{P_{g_{i+1}}\} = \{P_{g_i}\} + \{\Delta P_{g_i}\} , \tag{2}$$

$$\{q_{g_{i+1}}\} = \{q_{g_i}\} + \{\Delta q_{g_i}\} . \tag{3}$$

4.53.3 DMAP Calling Sequence

```
PLA2    DELTAUGV,DELTAPG,DELTAQG/UGV1,PGV1,QGL/V,N,PLACØUNT  $
```

4.53.4 Input Data Blocks

DELTAUGV - Incremental displacement vector in Piecewise Linear Analysis - g set.

DELTAPG - Incremental load vector in Piecewise Linear Analysis - g set.

DELTAQG - Incremental vector of single-point forces of constraint in Piecewise Linear Analysis - g set.

Note:

1. DELTAUGV and DELTAPG cannot be pre-purged.
2. DELTAQG may be pre-purged.

4.53.5 Output Data Blocks

UGV1 - Matrix of successive sums of incremental displacement vectors - g set.

PGV1 - Matrix of successive sums of incremental load vectors - g set.

QG1 - Matrix of successive sums of incremental vectors of single-point forces of constraint - g set.

Notes:

1. UGV1 and PGV1 cannot be purged.
2. QG1 may be purged if DELTAQG is purged.

4.53.6 Parameters

PLACØUNT - Input and output-integer - this parameter must be set to 1 outside the Piecewise Linear Analysis Rigid Format DMAP loop. This is done using the PARAM module rather than through the Module Properties List (MPL).

4.53.7 Method

If PLACØUNT = 1, that is, this is the first time PLA2 has been called in the Piecewise Linear Analysis Rigid Format DMAP loop, then the DELTAUGV data block is copied onto the UGV1 data block. If PLACØUNT > 1, then PLACØUNT is used as a counter to determine how many records (running sum displacement vectors) to skip on the file containing UGV1 so that the most recently computed running sum displacement vector can be read into open core for the vector addition. Once this vector is read into open core, the incremental displacement vector is read and interpreted using subroutines INTPK and ZNTPKI, and the vector addition given in Equation 1 is carried out element-by-element.

Equations 2 and 3 are computed using the method described in the above paragraph.

4.53.8 Subroutines

PLA2 has no auxiliary subroutines.

4.53.9 Design Requirements

Open core is defined at /PLA2X/.

4.53.10 Diagnostic Messages

User message 2127 or 2128 is output if either DELTAUGV (DELTAPG) or UGV1 (PGV1) is purged.

4.54 FUNCTIONAL MODULE PLA3 (PIECEWISE LINEAR ANALYSIS - PHASE 3)

4.54.1 Entry Point: PLA3

4.54.2 Purpose

To compute element stresses for nonlinear elements (see definition of linear elements in section 4.52.2) for which the user has requested stress output. It also updates the ESTNL data block so that the output data block, ESTNL1, contains up-to-date element stress information.

4.54.3 DMAP Calling Sequence

```
PLA3    CSTM,MPT,DIT,DELTAUGV,ESTNL,CASECC/ØNLES,ESTNL1/V,N,PLACØUNT/V,N,PLSETNØ  $
```

4.54.4 Input Data Blocks

CSTM - Coordinate System Transformation Matrices.

MPT - Material Properties Table.

DIT - Direct Input Tables.

DELTAUGV - Current incremental displacement vector.

ESTNL - Element Summary Table for Nonlinear Elements.

CASECC - Case Control Data Table.

Notes:

1. CSTM can be purged. However, if some grid point of the model is not in basic coordinates and the CSTM is purged, a fatal error occurs.

2. A fatal error occurs if either MPT, DIT, DELTAUGV, ESTNL or CASECC is purged.

4.54.5 Output Data Blocks

ØNLES - Nonlinear element stresses (to be processed by the Output File Processor).

ESTNL1 - Element Summary Table for Nonlinear Elements - Updated.

Note: Neither output data block may be purged.

4.54.6 Parameters

PLACØUNT - Input-integer-no default value. This is the Piecewise Linear Analysis (PLA)

Rigid Format DMAP loop counter. It is used in this routine to find the proper loading factors on the PLFACT bulk data card specified by the user (see PLSETNØ below).

PLSETNØ - Input-integer-no default value. PLSETNØ is the set identification number of some PLFACT bulk data card chosen by the user in his Case Control Deck. It is used to find this PLFACT card in the MPT data block.

4.54.7 Method

The module driver, PLA3, is a short routine whose only function is to call subroutines PLA31 and PLA32 which accomplish phase 1 and phase 2 of the task of the module respectively. Subroutine PLA31 reads the incremental displacement vector into core and appends to each element entry of the ESTNL data block the components of the incremental displacement vector corresponding to the grid points of each element. This merged information is written on the scratch data block ESTNLS, GINØ file number 301. In PLA32, the ESTNLS data block is read, and the proper element routine is called to compute element stresses which are prepared in ØFP (Output File Processor) format. Each element routine also updates incremental stress data. The ESTNL data for each element with the updated stress information (but without the components of the displacement vector) are written on ESTNL1.

In PLA31, for RØD, TUBE and BAR elements all six components of the displacement vector at each grid point of the element are appended to the ESTNL entry. Other elements for which Piecewise Linear Analysis is defined use only the three translational components at each grid point.

In PLA32, the difference quotients γ^* and γ, which are the previous and current (with respect to the DMAP loop in the PLA Rigid Format) load increment ratios, are computed as follows. Let $P_1, P_2, P_3, \ldots$, be the loading factors on a PLFACT bulk data card. Define $P_0 = 0$. Define

$$\alpha_i = P_i - P_{i-1} , \tag{1}$$

for $i \geq 1$. Then, define $\gamma_1^* = 0$, and

$$\gamma_i^* = \frac{\alpha_i}{\alpha_{i-1}} , \tag{2}$$

for $i > 1$, and define

$$\gamma_i = \frac{\alpha_{i+1}}{\alpha_i}, \qquad (3)$$

for $i \geq 1$. These difference quotients are stored in /PLA32C/ for communication to the module's element routines so that they can compute the estimated next strain. The details of the element calculation are given in section 4.87. The input parameter PLACØUNT, being the counter for the PLA Rigid Format DMAP loop, controls the computation of γ^* and γ. However, the module's design assumes (1) PLACØUNT is set to one outside the PLA DMAP loop and (2) module PLA2, which increments PLACØUNT by one, will be executed prior to every DMAP call to PLA3. Hence, the proper choice for the subscript i in Equations 1, 2 and 3 is one less than the value of PLACØUNT. The difference PLACØUNT-1 is stored in /PLA32C/ as IPASS.

4.54.8 Subroutines

PLA3 uses, for element routine calculations, the utility routines PRETRS, PREMAT, GMMATS and element drivers. Communication of an appended ESTNL element entry to an element routine during phase 2 of PLA3 is accomplished via /PLA32E/, which is 100 words in length. This fact is not explicitly stated below.

The element drivers PSTRM, PSQDM, PSTRI1, PSTRI2, PSQAD1, and PSQAD2, use a) /PLA3ES/, which is 300 words in length, as a communication link for the element subroutines which they call; and b) /PLA3UV/, which is 25 words in length, as a communication link for displacement vectors between the driver and their subroutines. PLA32 will call the element drivers listed above (plus PSRØD and PSBAR); the other subroutines described below (in sections 4.54.8.11 through 4.54.8.18) are only used (directly or indirectly) by the element drivers.

4.54.8.1 Subroutine Name: PLA31

1. Entry Point: PLA31
2. Purpose: To perform phase 1 of the module's operations as described above.
3. Calling Sequence: CALL PLA31

4.54.8.2 Subroutine Name: PLA32

1. Entry Point: PLA32

2. Purpose: To perform phase 2 of the module's operation as described above.

3. Calling Sequence: CALL PLA32

4.54.8.3 Subroutine Name: PSRØD

1. Entry Point: PSRØD

2. Purpose: To compute element stresses and to update the ESTNL entry for a RØD, CØNRØD or TUBE element. Note that for a TUBE element, the ESTNL entry is rearranged and elementary transformations are performed in PLA32 so that the PSRØD routine may compute element stresses for a TUBE.

3. Calling Sequence: CALL PSRØD

4.54.8.4 Subroutine Name: PSBAR

1. Entry Point: PSBAR

2. Purpose: To compute element stresses and to update the ESTNL entry for a BAR element.

3. Calling Sequence: CALL PSBAR

4.54.8.5 Subroutine Name: PSTRM

1. Entry Point: PSTRM

2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the TRMEM element.

3. Calling Sequence: CALL PSTRM

4.54.8.6 Subroutine Name: PSQDM

1. Entry Point: PSQDM

2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the QDMEM element.

3. Calling Sequence: CALL PSQDM

4.54.8.7 Subroutine Name: PSTRI1

1. Entry Point: PSTRI1

2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the TRIA1 element.

3. Calling Sequence: CALL PSTRI1

4.54.8.8 Subroutine Name: PSTRI2

1. Entry Point: PSTRI2

2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the TRIA2 element.

3. Calling Sequence: CALL PSTRI2

4.54.8.9 Subroutine Name: PSQAD1

1. Entry Point: PSQAD1

2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the QUAD1 element.

3. Calling Sequence: CALL PSQAD1

4.54.8.10 Subroutine Name: PSQAD2

1. Entry Point: PSAQD2

2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the QUAD2 element.

3. Calling Sequence: CALL PSQAD2

4.54.8.11 Subroutine Name: PSTRM1

1. Entry Point: PSTRM1

2. Purpose: To generate element stress matrices for the TRMEM element, and the membrane portion of TRIA1 and TRIA2 elements, and perform subcomputations for the PSQDM1 routine.

3. Calling Sequence: CALL PSTRM1 (NTYPE)

NTYPE { 0 = TRMEM, TRIA1, or TRIA2
1 = Subcomputations for the PSQDM1 subroutine

4.54.8.12 Subroutine Name: PSQDM1

1. Entry Point: PSQDM1

2. Purpose: To generate element stress matrices for the QDMEM element and the membrane portions of QUAD1 and QUAD2 elements.

3. Calling Sequence: CALL PSQDM1

4.54.8.13 Subroutine Name: PSTQ1

1. Entry Point: PSTQ1

2. Purpose: To generate element stress matrices for the TRIA1, TRIA2, QUAD1, and QUAD2 elements.

3. Calling Sequence: CALL PSTQ1 (NTYPE)

NTYPE { 1 = TRIA1
2 = TRIA2
3 = QUAD1
4 = QUAD2

4.54.8.14 Subroutine Name: PSTRB1

1. Entry Point: PSTRB1

2. Purpose: To generate element stress matrices for subcalculations of basic bending triangles for the plate portion of TRIA1, TRIA2, QUAD1 and QUAD2 elements.

3. Calling Sequence: CALL PSTRB1 (IØPT)

IØPT { 1 = Subcalculations for PSQPL1
2 = Subcalculations for PSTPL1

4.54.8.15 Subroutine Name: PSTPL1

1. Entry Point: PSTPL1

2. Purpose: To generate the element stress matrices for the plate portion of TRIA1 and

TRIA2 elements.

3. Calling Sequence: CALL PSTPL1

4.54.8.16 Subroutine Name: PSQPL1

1. Entry Point: PSQPL1

2. Purpose: To generate element stress matrices for the QUAD1 and QUAD2 elements.

3. Calling Sequence: PSQPL1

4.54.8.17 Subroutine Name: PSTRQ2

1. Entry Point; PSTRQ2

2. Purpose: To perform final stress computations for TRMEM and QDMEM elements.

3. Calling Sequence: CALL PSTRQ2 (NTYPE)

NTYPE { 1 = TRMEM element
      { 2 = QDMEM element

4.54.8.18 Subroutine Name: PSTQ2

1. Entry Point: PSTQ2

2. Purpose: To perform final stress computations for the TRIA1, TRIA2, QUAD1, and QUAD2 elements.

3. Calling Sequence: CALL PSTQ2 (NPTS)

NPTS { 3 = TRIA1 and TRIA2 elements
     { 4 = QUAD1 and QUAD2 elements

4.54.9 Design Requirements

1. The module was designed so that phase 1 and phase 2 can be executed in separate overlay segments.

2. Open core for phase 1 is defined at /PLA31X/ and for phase 2 at /PLA32X/. Open core requirements for both phases are minimal. In phase 1, the single precision incremental displacement vector in unpacked form must be able to be contained in open core. In phase 2, the CSTM and MPT data blocks, tables in the DIT referenced on MATS1 bulk data cards, and

the first record (and only record since a PLA problem allows only one CASECC record) of CASECC must be able to be contained in open core.

3. In addition to the common blocks mentioned above, PLA32 uses /PLA32S/, which is 325 words in length, as scratch storage for the module's element routines, and /SØUT/, which is 30 words in length, as a storage buffer for computed element stresses.

4. One scratch file is used, and all arithmetic operations are performed in single precision.

4.54.10 Diagnostic Messages

During phase 1, the following diagnostic messages may appear. If the incremental displacement vector is null, user fatal error 3005 will be given. Two system fatal "fail-safe" error messages, 2091 and 2092, may be implemented if the ESTNL input data block was incorrectly constructed in PLA1 or was incorrectly updated during the previous execution of the PLA3 module.

During phase 2, error messages 3001, 3002 or 3003 may occur if the proper loading factors P_j cannot be found on the PLFACT bulk data card image in the MPT. If the ECPTDS scratch file is not in the prescribed format, system fatal message 2091 will occur.

If the minimal core storage requirements in either phase 1 or phase 2 are not met, the usual fatal error 3008 will occur.

4.55 FUNCTIONAL MODULE PLA4 (PIECEWISE LINEAR ANALYSIS - PHASE 4)

4.55.1 Entry Point: PLA4

4.55.2 Purpose

To generate the stiffness matrix for nonlinear elements, $[K_{gg}^{n\ell}]$, and to update the Element Connection and Properties Table for Nonlinear Elements, ECPTNL, so that it contains up-to-date element stress information.

4.55.3 DMAP Calling Sequence

```
PLA4     CSTM,MPT,ECPTNL,GPCT,DIT,DELTAUGV/KGGNL,ECPTNL1/V,N,PLACØUNT/V,N,PLSETNØ/
         V,N,PLFACT   $
```

4.55.4 Input Data Blocks

CSTM - Coordinate System Transformation Matrices.

MPT - Material Properties Table.

ECPTNL - Element Connection and Properties Table for Nonlinear Elements.

GPCT - Grid Point Connection Table.

DIT - Direct Input Tables.

DELTAUGV - Current incremental displacement vector.

Notes:

1. CSTM may be purged. However, if some grid point of the model is not in basic coordinates and the CSTM has been purged, a fatal error will occur.
2. A fatal error occurs if either MPT, ECPTNL, GPCT, DIT or DELTAUGV is purged.

4.55.5 Output Data Blocks

KGGNL - Stiffness matrix of nonlinear elements - g set.

ECPTNL1 - Element Connection and Properties Table for Nonlinear Elements - updated.

Note: Neither KGGNL or ECPTNL1 may be purged.

4.55.6 Parameters

PLACØUNT - Input-integer-no default value. Loop counter for the Piecewise Linear Analysis (PLA) Rigid Format DMAP loop. The module uses this parameter to find the correct loading factors on the PLFACT bulk data card chosen by the user.

PLSETNØ - Input-integer-no default value. Set identification number of a PLFACT bulk data card chosen by the user in his Case Control Deck. The module uses this parameter to search the MPT for this card.

PLFACT - Output-complex-no default value. The difference of loading factors to be used during the next pass of the PLA Rigid Format DMAP loop.

4.55.7 Method

The module driver PLA4 is a short routine whose only function is to call subroutines PLA41 and PLA42 which accomplish phase 1 and phase 2 of the task of the module respectively. Subroutine PLA41 reads the incremental displacement vector into core and appends to each element entry of the ECPTNL data block the components of the incremental displacement vector corresponding to the grid points of each element. This merged information is written on the scratch data block ECPTS, GINØ file number 301. In PLA42, the ECPTS data block is processed in a fashion similar to the processing of the ECPT data block in module SMA1 (see the Module Functional Description for SMA1, section 4.27).

In PLA41, for all elements except the BAR element, only the three translational components of the displacement vector at each grid point of an element are appended to the ECPTNL element entry. For a BAR element, all six components of the displacement vector at each grid point are appended.

The logic of the processing of the scratch data block, ECPTS, in PLA42 is very similar to that used in subroutine SMA1A (of SMA1, the stiffness matrix generation module - see the Module Functional Description for SMA1, section 4.27). The similarities are not enumerated here, but notable differences are the following.

1. Before PREMAT is called to read into open core the MPT data block and tables from the DIT data block referenced on MATS1 bulk data cards, the MPT is read in subroutine

PLA42 to compute γ^* and γ as in Equations 1, 2 and 3 in section 4.54, and the real part of the output DMAP parameter PLFACT is set to the value of α_{i+1} in Equation 3 in section 4.54. The imaginary part of PLFACT is set to zero. The reason for PLFACT being complex is that it is an input parameter to the DMAP module ADD during the next pass of the PLA Rigid Format DMAP loop, and ADD (see section 4.78) requires its parameters to be complex.

2. When PREMAT is called, the last argument is set negative to signal PREMAT that this is a PLA problem and hence that special processing will be required.

3. Subsequent to the call of an element routine, the element type and the updated ECPT entry are written onto the ECPTNL1 data block.

4.55.8 Subroutines

PLA4 uses PRETRD, PRETRS, PREMAT, INVERS, INVERD, GMMATS, and GMMATD as utility routines. The common block /PLA42E/ is the means of communicating a) the element entry of the ECPTS from PLA42 to an element stiffness matrix generation routine and b) the ECPTS element entry with updated stress information from the element routine back to PLA42 upon completion of element matrix generation. This fact is not explicitly stated in the descriptions of the element routines (e.g., PKRØD) given below.

The element drivers PKTRM, PKQDM, PKTRI1, PKTRI2, PKQAD1, and PKQAD2 use a) /PLA4ES/, which is 300 words in length, and b) /PLA4UV/, which is 25 words in length, as communication links with the subroutines that they call. PLA42 will call the drivers listed above which will use (directly and indirectly) the subroutines described below in sections 4.55.8.12 through 4.55.8.22.

4.55.8.1 Subroutine Name: PLA41

1. Entry Point: PLA41

2. Purpose: See discussion above.

3. Calling Sequence: CALL PLA41

4.55.8.2 Subroutine Name: PLA42

1. Entry Point: PLA42

2. Purpose: See discussion above.

3. Calling Sequence: CALL PLA42

4.55.8.3 Subroutine Name: PLA4B

1. Entry Point: PLA4B

2. Purpose: To add a double precision 6 by 6 element stiffness matrix to the "submatrix" corresponding to the current pivot point. This routine performs the same function as, and is modeled after, subroutine SMA1B of module SMA1.

3. Calling Sequence: CALL PLA4B (KE,J)

KE - Row-stored double precision 6 by 6 matrix to be added to the submatrix in core - input.

J - The column index of the KGGNL matrix which corresponds to first column of the KE matrix - integer - input.

4.55.8.4 Subroutine Name: PKRØD

1. Entry Point: PKRØD

2. Purpose: To generate the element stiffness matrix for a RØD element and to update the ECPTNL element entry for a RØD element.

3. Calling Sequence: CALL PKRØD

4.55.8.5 Subroutine Name: PKBAR

1. Entry Point: PKBAR

2. Purpose: To generate the element stiffness matrix for a BAR element and to update the ECPTNL element entry for a BAR element.

3. Calling Sequence: CALL PKBAR

4.55.8.6 Subroutine Name: PKTRM

1. Entry Point: PKTRM

2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the TRMEM element.

3. Calling Sequence: PKTRM

4.55.8.7 Subroutine Name: PKQDM

1. EEntry Point: PKQDM

2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the QDMEM element.

3. Calling Sequence: CALL PKQDM

4.55.8.8 Subroutine Name: PKTRI1

1. Entry Point: PKTRI1

2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the TRIA1 element.

3. Calling Sequence: CALL PKTRI1

4.55.8.9 Subroutine Name: PKTRI2

1. Entry Point: PKTRI2

2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the TRIA2 element.

3. Calling Sequence: CALL PKTRI2

4.55.8.10 Subroutine Name: PKQAD1

1. Entry Point: PKQAD1

2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the QUAD1 element.

3. Calling Sequence: CALL PKQAD1

4.55.8.11 Subroutine Name: PKQAD2

1. Entry Point: PKQAD2

2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the QUAD2 element.

3. Calling Sequence: CALL PKQAD2

4.55.8.12 Subroutine Name: PKTRM1

1. Entry Point: PKTRM1

2. Purpose: To generate element stress matrices for the TRMEM, TRIA1 and TRIA2 elements, and perform subcomputations for the PKQDM1 routine.

3. Calling Sequence: CALL PKTRM1 (NTYPE)

NTYPE { 0 = TRMEM, TRIA1 or TRIA2
      { 1 = Subcomputations for the PKQDM1 routine

4.55.8.13 Subroutine Name: PKQDM1

1. Entry Point: PKQDM1

2. Purpose: To generate element stress matrices for the QDMEM, QUAD1 and QUAD2 elements.

3. Calling Sequence: CALL PKQDM1

4.55.8.14 Subroutine Name: PKTQ1

1. Entry Point: PKTQ1

2. Purpose: To generate element stress matrices for the TRIA1, TRIA2, QUAD1, and QUAD2 elements.

3. Calling Sequence: CALL PKTQ1 (NTYPE)

NTYPE { 1 = TRIA1
      { 2 = TRIA2
      { 3 = QUAD1
      { 4 = QUAD2

4.55.8.15 Subroutine Name: PKTRQ2

1. Entry Point: PKTRQ2

2. Purpose: To perform final stress computations for the TRMEM and QDMEM elements.

3. Calling Sequence: CALL PKTRQ2 (NTYPE)

NTYPE { 1 = TRMEM element
      { 2 = QDMEM element

4.55.8.16 Subroutine Name: PKTQ2

1. Entry Point: PKTQ2

2. Purpose: To perform final stress computations for the TRIA1, TRIA2, QUAD1, and QUAD2 elements.

3. Calling Sequence: CALL PKTQ2 (NPTS)

NPTS { 3 = TRIA1 or TRIA2 elements
     { 4 = QUAD1 or QUAD2 elements

4.55.8.17 Subroutine Name: PKTRMS

1. Entry Point: PKTRMS

2. Purpose: To generate the element stiffness matrix for the TRMEM element and sub-computations for the PKQDMS routine.

3. Calling Sequence: CALL PKTRMS (NTYPE)

NTYPE { 0 = TRMEM
      { 1 = Sub-computations for PKQDMS

4.55.8.18 Subroutine Name: PKQDMS

1. Entry Point: PKQDMS

2. Purpose: To generate the element stiffness matrix for the QDMEM element.

3. Calling Sequence: CALL PKQDMS

4.55.8.19 Subroutine Name: PKTRQD

1. Entry Point: PKTRQD

2. Purpose: To generate the element stiffness matrix for the TRIA1, TRIA2, QUAD1, or QUAD2 elements.

3. Calling Sequence: CALL PKTRQD (NTYPE)

NTYPE
- 1 = TRIA1
- 2 = TRIA2
- 3 = QUAD1
- 4 = QUAD2

4.55.8.20 Subroutine Name: PKTRBS

1. Entry Point: PKTRBS

2. Purpose: To generate the element stiffness matrix subcalculations for the PKTRPL and PKQDPL routines.

3. Calling Sequence: CALL PKTRBS (IØPT)

IØPT
- 1 = Subcomputations for PKQDPL
- 2 = Subcomputations for PKTRPL

4.55.8.21 Subroutine Name: PKTRPL

1. Entry Point: PKTRPL

2. Purpose: To generate the element stiffness matrix for the TRIA1 and TRIA2 elements.

3. Calling Sequence: CALL PKTRPL

4.55.8.22 Subroutine Name: PKQDPL

1. Entry Point: PKQDPL

2. Purpose: To generate the element stiffness matrix for the QUAD1 and QUAD2 elements.

3. Calling Sequence: CALL PKQDPL

4.55.9 Design Requirements

The module was designed so that phase 1 and phase 2 can be executed in separate overlay segments.

Open core for phase 1 is defined at /PLA41X/ and for phase 2 at /PLA42X/. In phase 1 the single precision incremental displacement vector in unpacked form must be able to be contained in core. In phase 2, the open core requirements are the same as those for module SMA1 (see section 4.27.9.1) except that only four GINØ buffers are required during the principal loop of phase 2,

which processes the ECPTS and GPCT in a complementary manner. One GINØ buffer is defined for each of KGGNL, ECPTNL1, ECPTS and GPCT.

In addition to /PLA42E/, which is 100 words in length, subroutine PLA42 uses the following common blocks: a) /PLA42D/, which is 300 double precision words in length, and is used as a scratch storage for the module's element routines; b) /PLA425/, which is 325 single precision words in length, and is used as scratch storage for the module element routines; and c) /PLA42C/, which is a communication region for phase 2 of the task of the module. /PLA42C/ is defined as follows:
CØMMØN/PLA42C/NPVT,GAMMA,GAMMAS,IPASS,ICSTM,NCSTM,IGPCT,NGPCT,IPØINT,NPØINT,I6X6K,N6X6K,CSTM,MPT,ECPTS,GPCT,DIT,KGGNL,ECPTØ,INRW,ØUTRW,EØR,NEØR,CLSRW,JMAX,FRØWIC,LRØWIC,NRØWSC,NLINKS,NWØRDS(40),IØVRLY(40),LINK(40),NØGØ

GAMMA,GAMMAS	-	The load increment ratios as defined in Equations 2 and 3 in section 4.54.
IPASS	-	Number of the current pass through the PLA DMAP loop.
NPVT,ICSTM,NCSTM,IGPCT, NGPCT,IPØINT,NPØINT, I6X6K,N6X6K	-	As defined in section 4.27.9.
CSTM,MPT,ECTPS,GPCT, DIT,KGGNL	-	GINØ file numbers for their corresponding data blocks.
ECPTØ	-	GINØ file number for the ECTPNL1 data block.
INRW,ØUTRW,..., IØVRLY(40),LINK(40),NØGØ	-	As defined in section 4.27.9.

The variables a) corresponding to GINØ file numbers, b) GINØ parameter options (e.g., INRW, ØUTRW), and c) NLINKS, IØVRLY, and NWØRDS, and NØGØ are set in the block data subprogram PLA4BD.

One scratch file is used, and all operations associated with stiffness matrix calculations are performed in double precision.

4.55.10 Diagnostic Messages

During phase 1, if the incremental displacement vector is null, user fatal error 2083 will occur.

During phase 2, error messages 3001, 3002, or 3003 may occur if the proper loading factors cannot be found on the PLFACT bulk data card image in the MPT. Other diagnostic messages for phase 2 are the same as those for module SMA1 (see section 4.27.10).

4.56 FUNCTIONAL MODULE CASE (SIMPLIFY CASE CONTROL)

4.56.1 Entry Point: CASE

4.56.2 Purpose

To remove looping considerations from later dynamics modules.

4.56.3 DMAP Calling Sequence

```
CASE    CASECC,PSDL/CASEXX/C,N,APPRØACH/V,N,REPEAT/V,N,LØØP  $
```

4.56.4 Input Data Blocks

CASECC - Case Control Data Table.

PSDL - Power Spectral Density List.

Note: PSDL is used only if APPRØACH = FREQRESP and Random Analysis is selected in CASECC.

4.56.5 Output Data Blocks

CASEXX - Case Control data table for dynamics problems.

Note: CASEXX cannot be purged.

4.56.6 Parameters

APPRØACH - Input-BCD-no default. Defines the approach to be used for looping criteria.

BCD Value	LØØP
STATICS	NØNE
REIGEN	NØNE
DSO	NØNE
DS1	NØNE
FREQRESP	DIRECT INPUT MATRICES OR TRANSFER FUNCTIONS
TRANRESP	LØADS
BLKO	NØNE
BLK1	NØNE
CEIGEN	DIRECT INPUT MATRICES OR TRANSFER FUNCTIONS

BCD Value	LØØP
PLA	NØNE

REPEAT - Input and output-integer-set equal to zero outside of the DMAP loop by the PARAM module. -1 if no additional loops; + loop count if loops.

LØØP - Output-integer-default = -1. -1 if this is not a looping problem, 0 if this is a looping problem.

4.56.7 Method

The method of operation depends upon the input parameter APPRØACH.

4.56.7.1 Transient Response

If APPRØACH = TRANRESP, CASECC is skipped over REPEAT records. If REPEAT = 0, REPEAT is set to 1. One record of CASECC is read and copied onto CASEXX. An attempt is made to read another record. If no more records exist, REPEAT is set to -1. Also, if this is the first entry to CASE (i.e., REPEAT = 1), LØØP is set to -1. If additional records exist, REPEAT and LØØP are set to 1.

4.56.7.2 Complex Eigenvalue Analysis

If APPRØACH = CEIGEN, REPEAT records are skipped in CASECC. If REPEAT = 0, REPEAT is set to 1. One record of CASECC is read and copied onto CASEXX. The names of the Direct Input Matrices and Transfer Functions sets are saved. An attempt is made to read another record. If no more exist, REPEAT is set to -1. Also if this is the first entry (i.e., REPEAT = 1) LØØP is set to -1. If additional records exist, their Direct Input Matrices and Transfer Functions sets are compared to those saved. If they all agree, this record is copied onto CASEXX and the process is repeated. If they do not agree, REPEAT is incremented by 1, LØØP is set to 1, and CASE returns.

4.56.7.3 Frequency Response

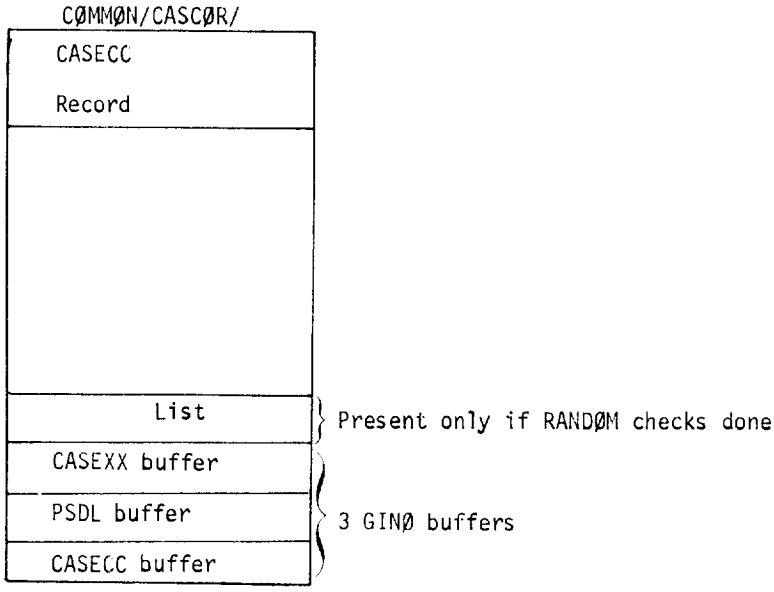
If APPRØACH = FREQRESP, the method used is the same as Complex Eigenvalue Analysis unless RANDPS cards are selected. If RANDPS cards are selected, the selected set is read from PSDL and the unique subcase "id's" referenced are stored. Each subcase id copied onto CASEXX is compared to this list, and the entry is marked as found. If at the completion of CASE unmarked entries exist, the routine terminates with message 3033.

4.56.8 Subroutines

No auxiliary subroutines are used by CASE.

4.56.9 Design Requirements

Open core is defined at /CASCØR/.



4.56.10 Diagnostic Messages

If a case control record cannot be held in core, CASE will issue error message 3008. Message 3033 may be issued by CASE as outlined above.

4.57 FUNCTIONAL MODULE MTRXIN (MATRIX INPUT)

4.57.1 Entry Point: MTRXIN

4.57.2 Purpose

MTRXIN has two purposes: (1) to provide a capability for direct input matrices as may occur in control systems in the dynamics Rigid Formats and, (2) to provide the DMAP user a capability of converting matrices input on DMIG bulk data cards to NASTRAN matrix format.

4.57.3 DMAP Calling Sequences

1. Dynamics Rigid Formats:

```
MTRXIN    CASECC,MATPØØL,EQDYN,SILD,TFPØØL/K2PP,M2PP,B2PP/V,N,LUSETD/V,N,NØMAT1/V,N,NØMAT2/
          V,N,NØMAT3  $
```

2. DMAP Approach:

```
MTRXIN,   ,MATPØØL,EQEXIN,SIL,/NAME1,NAME2,NAME3/V,N,LUSET/V,N,NØMAT1/V,N,NØMAT2/V,N,NØMAT3   $
```

4.57.4 Input Data Blocks

CASECC - Case Control.

MATPØØL - Data block containing matrices input on DMIG bulk data cards.

EQDYN - Equivalence between external numbers and internal numbers, dynamics.

SILD - Scalar Index List - dynamics.

TFPØØL - Transfer Function Pool.

EQEXIN - Equivalence between external numbers and internal numbers.

SIL - Scalar Index List.

Notes:

1. If CASECC is purged, the second purpose is assumed by MTRXIN.
2. EQDYN, EQEXIN, SIL and SILD may not be purged.

4.57.5 Output Data Blocks

K2PP - Direct input stiffness matrix - p set.

M2PP - Direct input mass matrix - p set.

B2PP - Direct input damping matrix - p set.

NAME1, NAME2, NAME3 - The same names that appear on the DMIG cards, i.e., the DMIG matrix called NAME1 will be output on data block NAME1.

Note: Any output data block may be purged.

4.57.6 Parameters

LUSET - Input-integer-no default. Degrees of freedom in the g set. Used with EQEXIN and SIL.

LUSETD - Input-integer-no default. Degrees of freedom in the p set. Used with EQDYN and SILD.

NØMATi - Output-integer-no default. +1 if the i^{th} output data block is generated, -1 otherwise.

4.57.7 DMAP Example

Assume the bulk data contain two DMIG matrices named M1 and M2 which reference grid and/or scalar points only. The following set of DMAP instructions will generate these two matrices in NASTRAN matrix format, multiply them together and print the result.

```
BEGIN
GP1       GEØM1,GEØM2/GPL,EQEXIN,GPDT,CSTM,BGPDT,SIL/V,N,LUSET/C,N,0/C,N,0  $
SAVE      LUSET $
MTRXIN,   ,MATPØØL,EQEXIN,SIL,/M1,M2,/V,N,LUSET/V,N,NØM1/V,N,NØM2/C,N,0 $
SAVE      NØM1,NØM2 $
CØND      EXIT,NØM1 $
CØND      EXIT,NØM2 $
MPYAD     M1,M2,/PRØDUCT/C,N,0/C,N,1/C,N,1/C,N,1 $
MATPRN    PRØDUCT,,,,// $
LABEL     EXIT $
END
```

4.57.8 Method

The first logical record in the Case Control data block is read into core, and the names of the requested DMIG matrices are fetched. If the Case Control data block is purged, FNAME is called to determine the names of the DMIG matrices from the names of the output data blocks. If the Case Control record was read, the transfer function set selection is fetched. If transfer matrices are requested, the TFPØØL data block is opened, and the file is positioned to the requested set. Each transfer function matrix for which a corresponding direct input matrix exists is written on a scratch file. If no direct input matrix exists corresponding to a transfer function matrix, the transfer function matrix is written directly on the appropriate output data block. The transfer function matrices are written in NASTRAN matrix format by decoding the row and column numbers and calling BLDPK.

Upon completion of the writing of the transfer function matrices (if any), the second record of EQEXIN or EQDYN is read into core. The second word of each entry is converted into a scalar index number by dividing by 10. The MATPØØL data block is opened. The following processing occurs:

1. The header information for the DMIG matrix is read. If an end-of-file is encountered, step (5) is executed. If the matrix is not requested, the remainder of the record is skipped and step (1) is repeated. Otherwise, step (2) is executed.

2. Each term in the matrix is read. The grid identification and component code are converted to a scalar index value by performing a binary search in EQEXIN or EQDYN in core. The scalar index forms a row position of the matrix. The row and column number (packed in one word) and the value for the term are stored in core. If core storage is exceeded, the terms are written on a scratch file.

3. When all terms have been read, converted and stored, the matrix is sorted by SØRT. The matrix is now written in NASTRAN format by BLDPK.

4. If a transfer function is to be added to the DMIG matrix, the ADD routine is called to accomplish the matrix addition.

5. A test is made to determine if all requested matrices have been processed. If not, an error message is queued, and PEXIT is called. Otherwise, the module makes a normal exit.

4.57.9 Design Requirements

4.57.9.1 Allocation of Core Storage

Storage is required to hold the EQDYN or EQEXIN table (2 words per point in the problem) plus five GINØ buffers. Complete spill logic is provided for processing the DMIG matrices.

4.57.9.2 Environment

The module MTRXIN consists of one subroutine, MTRXIN. Calls are made to the utility routine SØRT and matrix operation ADD. Open core is defined by /MTRXXX/. Seven scratch files are used.

4.57.10 Diagnostic Messages

The following messages may be issued by MTRXIN:

2065, 2070, 2074.

4.58 FUNCTIONAL MODULE GKAD (GENERAL K ASSEMBLER DIRECT)

4.58.1 Entry Point: GKAD

4.58.2 Purpose

To assemble the dynamic stiffness, damping and mass matrices.

4.58.3 DMAP Calling Sequence

```
GKAD  USETD,GM,GØ,KAA,BAA,MAA,K4AA,K2PP,M2PP,B2PP/KDD,BDD,MDD,GMD,GØD,K2DD,M2DD,B2DD/
      V,N,TYPE/V,N,APP/V,N,FØRM/V,Y,G/V,Y,W3/V,Y,W4/V,N,NØK2PP/V,N,NØM2PP/V,N,NØB2PP/
      V,N,MPCF1/V,N,SINGLE/V,N,ØMIT/V,N,NØUE/V,N,NØK4GG/V,N,NØBGG/V,N,KDEKA/V,Y,MØDACC   $
```

4.58.4 Input Data Blocks

USETD - Displacement set definitions table dynamics.

GM - Multipoint constraint transformation matrix - m set.

GØ - Structural matrix partitioning transformation matrix.

KAA - Partition of stiffness matrix - a set.

BAA - Partition of damping matrix - a set.

MAA - Partition of mass matrix - a set.

K4AA - Partition of structural damping matrix - a set.

K2PP - Direct input stiffness matrix - p set.

M2PP - Direct input mass matrix - p set.

B2PP - Direct input damping matrix - p set.

Notes:
1. USETD cannot be purged.
2. GM cannot be purged if MPCF1 ≥ 0.
3. GØ cannot be purged if ØMIT ≥ 0.
4. KAA cannot be purged if KDEKA ≥ 0.
5. BAA cannot be purged if NØBGG ≥ 0.
6. MAA may be purged.
7. K4AA cannot be purged if NØK4GG ≥ 0.
8. K2PP cannot be purged if NØK2PP ≥ 0.
9. M2PP cannot be purged if NØM2PP ≥ 0.

10. B2PP cannot be purged if NØB2PP ≥ 0.

4.58.5 Output Data Blocks

KDD - Dynamic stiffness matrix - d set.

BDD - Dynamic damping matrix - d set.

MDD - Dynamic mass matrix - d set.

GMD - Multipoint constraint transformation matrix - dynamics.

GØD - Omitted coordinate transformation matrix - dynamics.

K2DD - Direct input stiffness matrix - d set.

M2DD - Direct input mass matrix - d set.

B2DD - Direct input damping matrix - d set.

Notes: 1. GMD cannot be purged if MPCF1 ≥ 0.

2. GØD cannot be purged if ØMIT ≥ 0.

3. K2DD cannot be purged if NØK2PP ≥ 0.

4. M2DD cannot be purged if NØM2PP ≥ 0.

5. B2DD cannot be purged if NØB2PP ≥ 0.

4.58.6 Parameters

TYPE - Input-BCD-no default. If TYPE = TRANSIENT the transient equations are used; otherwise the frequency response equations are used.

APP - Input-BCD-no default. If APP = FØRCE the p set = d set; otherwise p's are reduced to d's by removing m's, s's. and o's.

FØRM - Input-BCD-no default. If FØRM = MØDAL, KDD and BDD are not computed. MDD is not computed unless MØDACC ≥ 0.

G - Input-real-default = 0.0. G is the coefficient of K4DD if TYPE ≠ TRANSIENT. G/W3 is coefficient of K1DD if TYPE = TRANSIENT.

W3 - Input-real-default = 0.0. If TYPE = TRANSIENT G/W3 is the coefficient of K1DD. If W3 = 0.0 K1DD is not used.

W4 - Input-real-default = 0.0. 1.0/W4 is the coefficient of K4DD if TYPE = TRANSIENT. If W4 = 0.0 K4DD is not used.

NØK2PP - Input-integer-no default. NØK2PP ≥ 0 indicates presence of K2PP.

NØM2PP - Input-integer-no default. NØM2PP ≥ 0 indicates presence of M2PP.

NØB2PP - Input-integer-no default. NØB2PP ≥ 0 indicates presence of B2PP.

MPCF1 - Input-integer-no default. MPCF1 ≥ 0 indicates presence of GM.

SINGLE - Input-integer-no default. SINGLE ≥ 0 indicates presence of single-point constraints.

ØMIT - Input-integer-no default. ØMIT ≥ 0 indicates presence of GØ.

NØUE - Input-integer-no default. NØUE ≥ 0 indicates presence of extra points.

NØK4GG - Input-integer-no default. NØK4GG ≥ 0 indicates presence of K4AA.

NØBGG - Input-integer-no default. NØBGG ≥ 0 indicates presence of BAA.

KDEKA - Input-integer-no default. KDEKA ≥ 0 indicates presence of MAA and KAA.

MØDACC - Input-integer-default = -1. MØDACC ≥ 0 requests computation of MDD (meaningful only if FØRM = MØDAL)

4.58.7 Method

If extra points are present (NØUE≥0) and multipoint constraints or omitted coordinates are present (MPCF1≥0 or ØMIT≥0), then

$$GM \Longrightarrow GMD, \tag{1}$$

and

$$G\emptyset \Longrightarrow G\emptyset D. \tag{2}$$

Subroutine GKAD1A performs these tasks.

If direct input matrices are present and m's, s's or o's are present, the direct input matrices are reduced from the p set to the d set. Let $[D^2_{pp}]$ be a direct input matrix,

$$[D^2_{pp}] = [K^2_{pp}], [M^2_{pp}] \text{ or } [B^2_{pp}]$$

1. If m's are present,

$$[D^2_{pp}] \Longrightarrow \left[\begin{array}{c|c} \bar{D}^2_{nn} & \bar{D}^2_{nm} \\ \hline \bar{D}^2_{mn} & \bar{D}^2_{mm} \end{array}\right] . \qquad (3)$$

(The e coordinates are included with the n coordinates). Then compute:

$$[D^2_{nn}] = [\bar{D}^2_{nn}] + [\bar{D}^2_{nm}]\,[G^d_m] + [G^d_m]^T\,[\bar{D}^2_{mn}] + [G^d_m]^T\,[\bar{D}^2_{mn}]\,[G^d_m] .$$

2. If s's are present,

$$[D^2_{ff}] \Longrightarrow \left[\begin{array}{c|c} D^2_{ff} & D^2_{fs} \\ \hline D^2_{sf} & D^2_{ss} \end{array}\right] , \qquad (5)$$

where only $[D^2_{ff}]$ is saved. The e coordinates are included with the f coordinates.

3. If o's are present, first partition $[D^2_{ff}]$

$$[D^2_{ff}] \Longrightarrow \left[\begin{array}{c|c} \bar{D}^2_{dd} & \bar{D}^2_{do} \\ \hline \bar{D}^2_{od} & \bar{D}^2_{oo} \end{array}\right] , \qquad (6)$$

then:

$$[D^2_{dd}] = [\bar{D}^2_{dd}] + [\bar{D}^2_{do}]\,[G^d_o] + [G^d_o]^T\,[D_{od}] + [G^d_o]^T\,[\bar{D}^2_{oo}]\,[G^d_o]. \qquad (7)$$

Steps 1 through 3 are done for K2PP, M2PP and B2PP, using subroutines GKAD1C and GKAD1D.

If FØRM = MØDAL and MØDACC < 0, GKAD is done. If not, the a set matrices are expanded to the d set by adding zeros at extra points. Let $[D_{aa}]$ be an a set matrix. Then,

$$\begin{bmatrix} D_{aa} & 0 \\ 0 & 0 \end{bmatrix} \Longrightarrow \quad [D^{1}_{dd}] \ . \tag{8}$$

The above step is done for KAA, BAA, MAA, and K4AA and is performed in subroutine GKAD1B.

Compute KDD, BDD and MDD.

1. For Frequency Response or Complex Eigenvalue Analysis (TYPE ≠ TRAN),

$$[K_{dd}] = (1+iG)\,[K^{1}_{dd}] + [K^{2}_{dd}] + i\,[K^{14}_{dd}] \ , \tag{9}$$

$$[B_{dd}] = [B^{1}_{dd}] + [B^{2}_{dd}] \ , \tag{10}$$

$$[M_{dd}] = [M^{1}_{dd}] + [M^{2}_{dd}] \ . \tag{11}$$

2. For Transient Analysis (TYPE = TRAN),

$$[K_{dd}] = [K^{1}_{dd}] + [K^{2}_{dd}] \ , \tag{12}$$

$$[B_{dd}] = [B^{1}_{dd}] + [B^{2}_{dd}] + \frac{G}{W3}\,[K^{1}_{dd}] + \frac{1.0}{W4}\,[K^{14}_{dd}] \ , \tag{13}$$

$$[M_{dd}] = [M^{1}_{dd}] + [M^{2}_{dd}] \ . \tag{14}$$

If W3 or W4 is zero, the corresponding matrices are ignored.

4.58.8 Subroutines

GKAD uses matrix utility routines SSG2C, CALCV, MERGE, UPART, MPART and ELIM. Descriptions for these routines can be found in section 3.

4.58.8.1 Subroutine Name: GKAD1A

1. Entry Point: GKAD1A

2. Purpose: To expand GM or GØ to d size matrices:

$$[G_m \vdots o] \Longrightarrow [G_m^d] \tag{15}$$

3. Calling Sequence: CALL GKAD1A (USETD,GØ,GØD,SCR1,UE,UA,UNE)

USETD - GINØ file number of USETD - integer - input.

GØ - GINØ file number of GØ - integer - input.

GØD - GINØ file number of GØD - integer - input.

SCR1 - GINØ file number of scratch file - integer - input.

UE - Pointer to UE bit in USETD word - integer - input.

UA - Pointer to UA bit in USETD word - integer - input.

UNE - Pointer to UNE bit in USETD word - integer - input.

4.58.8.2 Subroutine Name: GKAD1B

1. Entry Point: GKAD1B

2. Purpose: To expand a set matrices to d set size.

3. Calling Sequence: CALL GKAD1B (USETD,KAA,MAA,BAA,K4AA,K1DD,M1DD,B1DD,K41DD,UA, UE,UD,SCR1)

USETD - GINØ file number of USETD - integer - input.

KAA - GINØ file number of KAA - integer - input.

MAA - GINØ file number of MAA - integer - input.

BAA - GINØ file number of BAA - integer - input.

K4AA - GINØ file number of K4AA - integer - input.

K1DD - GINØ file number of K1DD - integer - input.

M1DD - GINØ file number of M1DD - integer - input.

B1DD - GINØ file number of B1DD - integer - input.

K41DD - GINØ file number of K41DD - integer - input.

SCR1 - GINØ file number of scratch file - integer - input.

UA - Pointer to UA bit in USETD word - integer - input.

UE - Pointer to UE bit in USETD word - integer - input.

UD - Pointer to UD bit in USETD word - integer - input.

4.58.8.3 Subroutine Name: GKAD1C

1. Entry Point: GKAD1C

2. Purpose: To initialize GKAD1D.

3. Calling Sequence: CALL GKAD1C (GMD,GØD,SCR1,SCR2,SCR3,SCR4,SCR5,SCR6,USETD)
   GMD,GØD,USETD are GINØ file numbers of their respective data blocks - integer - input.
   SCR1,...,SCR6 are GINØ file numbers of six scratch files - integer - input.

4.58.8.4 Subroutine Name: GKAD1D

1. Entry Point: GKAD1D

2. Purpose: To reduce "2PP" matrices to "2DD" matrices.

3. Calling Sequence: CALL GKAD1D (K2PP,K2DD)
   K2PP - GINØ file number of input matrix - integer - input.
   K2DD - GINØ file number of reduced matrix - integer - input.

4.58.9 Design Requirements

Six scratch files are necessary. Open core for GKAD1A and GKAD1B is defined at /GKADA1/. Open core for GKAD1C and GKAD1D is defined at /GKADC1/.

4.58.10 Diagnostic Messages

None

4.59 FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

4.59.1 Entry Point: CEAD

4.59.2 Purpose

To solve the equation

$$\left([M]p^2 + [B]p + [K]\right)\{u\} = \{0\} \qquad (1)$$

for the eigenvalues p and the associated eigenvectors {u} where [M], [B] and [K] are mass, damping and stiffness matrices respectively.

4.59.3 DMAP Calling Sequence

```
CEAD     KDD,BDD,MDD,EED,CASECC/PHID,CLAMA,ØCEIGS/V,N,NFØUND   $
```

4.59.4 Input Data Blocks

KDD - Dynamic stiffness matrix - d set.

BDD - Dynamic damping matrix - d set.

MDD - Dynamic mass matrix - d set.

EED - Eigenvalue Extraction Data.

CASECC - Case Control Data Table.

Notes:

1. EED must be present.
2. CASECC must be present.
3. At least one of KDD, BDD and MDD must be present.

4.59.5 Output Data Blocks

PHID - Complex eigenvectors in the d set.

CLAMA - Complex eigenvalue table.

ØCEIGS - Complex eigenvalue summary table.

Note: No output data block can be purged.

4.59.6 Parameters

NFØUND - Output-integer-no default. NFØUND indicates the number of eigenvalues found. If none were found, NFØUND is set to -1.

4.59.7 Method

The Complex Eigenvalue Analysis Module calculates the eigenvalues and eigenvectors for a general system which may have complex terms in the mass, damping, and stiffness matrices. The eigenvectors are scaled according to the user-requested normalization scheme. Modal masses are not calculated since they will, in general, be complex, and their value is rather dubious. The form of the problem solved by the Complex Eigenvalue Analysis Module is given in Equation 1.

The eigenvalues p and the eigenvectors {u} are always treated as complex. These data are related to the u_d displacements if a direct formulation is used or are related to the u_h displacements if a modal formulation is used. The method to be used and the necessary data are selected by calling for one ID number in the EED data block. A set of EED data which defines either the Determinant Method or the Inverse Power Method must be used. Subroutine CINVPR or CDETM is called to solve the eigenvalue problem (see subroutine descriptions below for method details). The eigenvalues and associated vectors are sorted by the magnitude of the imaginary part of the eigenvalue with all positives listed ahead of all negatives. (Subroutine CEAD1A).

4.59.8 Subroutines

The subroutines used by CEAD can be divided into four groups: 1) those used by CEAD; 2) those used for the Inverse Power Method; 3) those used by the Determinant Method; and, 4) general utility routines. The descriptions of the utility routines can be found in section 3.

CEAD	Inverse Power		Determinant	General
CEAD1A	CINVPR	CINFBS	CDETM	CDCØMP
	CINVP1	CMTIMU	CDETM2	ADD
	CINVP2	CXTRNY	CSUMM	PRELØC
	CINVP3	CSUB	CSQRT	
	CNØRM	ØRTHØ	CDTFBS	
	CNØRM1	CDIFBS	CDETM3	
		CSQRTX		

4.59.8.1 Subroutine Name: CEAD1A

1. Entry Point: CEAD1A

2. Purpose: To sort the eigenvectors and eigenvalues.

3. Calling Sequence: CALL CEAD1A (LAMAI,PHII,LAMAØ,PHIØ,NFØUND)

LAMAI - GINØ file number of unsorted eigenvalues - integer - input.

PHII - GINØ file number of unsorted eigenvectors - integer - input.

LAMAØ - GINØ file number of data block CLAMA - integer - input.

PHIØ - GINØ file number of data block PHID - integer - output.

NFØUND - Number of eigenvalues found - integer - input.

4.59.8.2 Subroutine Name: CINVPR

1. Entry Point: CINVPR

2. Purpose: CINVPR is the main driver for the Complex Inverse Power Method of eigenvalue extraction.

3. Calling Sequence: CALL CINVPR (EED,METHØD,NFØUND)
CØMMØN /CINVPX/K(7),M(7),B(7),LAM(7),PHI(7),EIGSUM,SCRFIL(11),NØREG,EPS,REG(7,10)
CØMMØN /CINVX/Z(1)

K,M,B - Input matrix control blocks for the stiffness, mass, and damping matrices [K], [M], and [B].

LAM,PHI - Matrix control blocks for the output eigenvalue and eigenvector files.

EIGSUM - The output eigenvalue summary file.

SCRFIL(11) - Eleven scratch files available to Inverse Power.

NØREG - Number of regions input to CINVPR.

EPS - Convergence criterion.

REG(7,10) - Storage space for up to 10 region parameters.

Z(1) - Open core for CINVPR.

EED - GINØ file number for this input data block.

METHØD - ID of an EIGC card for the Inverse Power Method.

NFØUND - Number of eigenvalues found.

4. Method: Complex Inverse Power was, in general, designed identically to Real Inverse Power. The notable exceptions are in the iteration equation and the orthogonalization with respect to previously extracted eigenvectors. With these points in mind, the basic flow can be taken from READ in section 4.48. Theoretical development is given in the Theoretical Manual.

4.59.8.3 Subroutine Name: CINVP1

1. Entry Point: CINVP1

2. Purpose: To generate calling sequences to ADD to form

$$[A] = [K] + \lambda[B] + \lambda^2[M]. \qquad (2)$$

3. Calling Sequence: CALL CINVP1

```
CØMMØN /CINVPX/ K(7),M(7),B(7),DUM(15),A
CØMMØN /CINV1X/Z(1)
CØMMØN /CINVXX/ LAMBDA
```

K,M,B - Matrix control blocks for the input matrices.

A - GINØ file number for the output matrix.

Z(1) - Area of open core available to ADD.

LAMBDA - Complex double precision scalar multiplier.

4.59.8.4 Subroutine Name: CINVP2

1. Entry Point: CINVP2

2. Purpose: To initialize and call CDCØMP for subroutine CINVPR.

3. Calling Sequence: CALL CINVP2

```
COMMON /CINVPX/ DUM(36),A,XX,L,U,SCR1,SCR2,SCR3,LL,UU
COMMON /CINVXX/DUMM(4),SWITCH
COMMON /CINV2X/Z(1)
```

A	-	GINO file number for the input matrix.
L,U	-	GINO file number for the lower and upper triangular factors output from CDCOMP.
SCR1,SCR2,SCR3	-	Three scratch files used by CDCOMP.
LL,UU	-	GINO file numbers for alternate storage of L and U.
SWITCH	-	0, store factors on L and U. 1, store factors on LL and UU.
Z(1)	-	Area of open core used by CDCOMP.

4.59.8.5 Subroutine Name: CINVP3

1. Entry Point: CINVP3

2. Purpose: To solve for a complex eigenvalue and eigenvector using the Inverse Power Method.

3. Calling Sequence: CALL CINVP3

CØMMØN /CINVPX/K(7),M(7),B(7),LAM(7),PHI(7),XXX,SCRFIL(11)

CØMMØN /CINV3X/Z(1)

See section 4.59.8.2 above for details on /CINVPX/.

Z(1) - Area of open core available in CINVP3.

4. Method: The logic flow and the mathematical equations are essentially identical to INVP3, with the following exceptions. The eigenvalues and eigenvectors are found corresponding to the matrix equation

$$(\lambda^2[M] + \lambda[B] + [K])\ [\Phi] = [0] \tag{3}$$

where the iteration equation is given by

$$(\lambda_o^2[M] + \lambda_o[B] + [K])\ \{W_n\} = -([B] + \lambda_o[M])\ \{u_{n-1}\} - [M]\ \{v_{n-1}\} \tag{4}$$

with

$$\{\bar{u}_n\} = \frac{1}{C_n}\{W_n\}\ , \tag{5}$$

$$\{\bar{v}_n\} = \lambda_o\{\bar{u}_n\} + \frac{1}{C_n}\ \{u_{n-1}\}\ , \tag{6}$$

$$\{u_n\} = \{\bar{u}_n\} - \sum_i \alpha_i\{\Phi_i\}, \tag{7}$$

$$\{v_n\} = \{\bar{v}_n\} - \sum_i \alpha_i\ \lambda_i\{\Phi_i\}, \tag{8}$$

and

$$\alpha_i = \frac{\{\Phi_i\}^T\ [\lambda_i[M]\{\bar{u}_n\} + [M]\ \{\bar{v}_n\} + [B]\ \{\bar{u}_n\}]}{\{\Phi_i\}^T\ (2\lambda_i\ [M] + [B])\ \{\Phi_i\}}\ . \tag{9}$$

where

Φ_i = Previously extracted right-hand vector,

$\bar{\Phi}_i$ = Previously extracted left-hand vector,

C_n = Largest element (in magnitude) of $\{W_n\}$, and

λ_1 = Previously extracted eigenvalue.

The above equations replace Equations 19 through 22 in section 4.48. The calculation of the remaining equations remains the same except for the use of complex arithmetic. The left eigenvector is obtained by decomposing Equation 3 with $\lambda_o - \lambda_i$ and using CDIFBS to make the appropriate substitution using the factors from CDCØMP.

5. Design Requirements: CINVPS requires fourteen complex double precision vectors in core plus four GINØ buffers.

4.59.8.6 Subroutine Name: CNØRM

1. Entry Point: CNØRM

2. Purpose: To normalize successive iterated vectors such that the maximum element is equal to unity, and to return the normalizing divisor.

3. Calling Sequence: CALL CNØRM (X,DIV)

X - Input vector to be normalized.

DIV - Divisor which was used to normalize the vector corresponding to the argument X.

4.59.8.7 Subroutine Name: CNØRM1

1. Entry Point: CNØRM1

2. Purpose: To normalize a complex vector such that the largest magnitude of an element is equal to one.

3. Calling Sequence: CALL CNØRM (X,N)

X - Vector to be normalized.

N - Length of the vector (complex terms).

4.59.8.8 Subroutine Name: CINFBS

1. Entry Point: CINFBS

2. Purpose: To perform the forward-backward substitution necessary to solve an iteration of the Inverse Power Method.

3. Calling Sequence: CALL CINFBS (X,Y,BUF)

CØMMØN /CINFBX/L(7),U(7)

L,U - Matrix control blocks for the factors output from CDCØMP.

X - Complex double precision input vector.

Y - Complex double precision solution vector.

BUF - GINØ buffer.

4. Method: CINFBS is a stripped down version of GFBS. Both vectors reside in core, and only complex double precision arithmetic is used.

4.59.8.9 Subroutine Name: CDIFBS

1. Entry Point: CDIFBS

2. Purpose: To perform the forward-backward substitution necessary to solve for the left eigenvector.

3. Calling Sequence: CALL CDIFBS (X,BUF)

CØMMØN /CINVPX/DUM(41),UPRTRI,XXX,LØWTRI

UPRTRI,LØWTRI - Files containing the upper and lower triangular factors output from CDCØMP.

X - The output complex double precision left eigenvector.

BUF- GINØ buffer used by CDIFBS.

4. Method: CDIFBS actually solves the system of equations

$$[A]^T \{x\} = \{y\}, \tag{10}$$

where [A] has been decomposed into [A] = [L] [U]. To solve the transpose problem we have that

$$[A]^T = ([L]\,[U])^T = [U]^T\,[L]^T \tag{11}$$

so that

$$[U]^T\,[L]^T\,\{x\} = \{y\}. \tag{12}$$

CDIFBS is a modified form of GFBS which does the forward pass on [U] and the backward pass on [L]. All arithmetic operations are complex double precision.

4.59.8.10 Subroutine Name: CMTIMU

1. Entry Point: CMTIMU

2. Purpose: To pre-multiply a vector {y} by a matrix to obtain a vector {x}.

3. Calling Sequence: CALL CMTIMU (Y,X,FILE,BUF)

CØMMØN /CINVPX/DUM(7),M(7)

FILE - If FILE = 0, form {x} = [M]{y}.
FILE ≠ 0, form {x} = [A] {y}, where [A] is the matrix on FILE.

X,Y - Complex double precision vectors.

BUF - GINØ buffer.

4.59.8.11 Subroutine Name: CXTRNY

1. Entry Point: CXTRNY

2. Purpose: To form the inner product of two complex vectors, {x} and {y}

$$a = \{x\}^T\,\{\bar{y}\}. \tag{13}$$

where $\{\bar{y}\}$ denotes a vector all of whose components are the complex conjugates of {y}.

3. Calling Sequence: CALL CXTRNY (X,Y,A)

CØMMØN /CINVPX/XX,N

N - Length of the vectors.

X,Y - Complex double precision vectors.

A - Complex double precision value of the inner product of {x} and {y}.

4.59.8.12 Subroutine Name: CSUB

1. Entry Point: CSUB

2. Purpose: To evaluate the vector equation

$$\{z\} = a\{x\} - b\{y\}, \tag{14}$$

where {x}, {y}, a and b may be complex.

3. Calling Sequence: CALL CSUB (X,Y,Z,A,B)

CØMMØN /CINVPX/XXX,N

N - Length of the vectors {x} and {y}.

X,Y,Z - Complex double precision vectors.

A,B - Complex double precision scalar multipliers.

4.59.8.13 Subroutine Name: ØRTHØ

1. Entry Point: ØRTHØ

2. Purpose: To orthogonalize a vector with respect to all previously extracted vectors.

3. Calling Sequence: CALL ØRTHØ (U,V,X1,X2,X3,X4,X5,NZ,BUF1,BUF2,BUF3,BUF4)

CØMMØN /CINVPX/K(7),M(7),B(7),LAMBDA(7),PHI(7),XXX,SCRFIL(10)

CØMMØN /CINVXX/DUM(19),NRØØTS

See section 4.59.8.2 for /CINVPX/ details.

NRØØTS - Number of eigenvectors already extracted.

U,V - Input-current vectors - Output - orthogonalized vectors.

X1,...,X5- Storage space for five complex double precision vectors.

NZ - The number of words of core available to ØRTHØ.

BUF1, ..., BUF4 - Four GINØ buffers.

4. Method: ØRTHØ solves the equations

$$\{u_n\} = \{u_n\} - \sum_i \alpha_i \lambda_i \{\Phi_i\}, \tag{15}$$

$$\{v_n\} = \{v_n\} - \sum_i \alpha_i \lambda_i \{\Phi_i\}, \tag{16}$$

where

$$\alpha_i = \frac{\{\bar{\Phi}_i\}^T[\lambda_i[M]\{u_n\} + [M]\{v_n\} + [B]\{u_n\}]}{\{\bar{\Phi}_i\}^T[2\lambda_i[M] + [B]]\{\Phi_i\}} \tag{17}$$

and

$\{\bar{\Phi}_i\}$ = Previously found left eigenvectors.

$\{\Phi_i\}$ = Previously found right eigenvectors.

λ_i = Previously found eigenvalues.

4.59.8.14 Subroutine Name: CDETM

1. Entry Point: CDETM

2. Purpose: To solve the complex eigenvalue problem by the Determinant Method.

3. Calling Sequence: CALL CDETM (METHØD,EED,M,B,K,LAMA,PHID,ØCEIGS,NFØUND,SCR1,SCR2, SCR3,SCR4,SCR5,SCR6,SCR7,SCR8)

METHØD - ID of an EIGC card for the Determinant Method - integer - input.

EED,ØCEIGS, M,B,K - GINØ file numbers of their respective data blocks - integer - input.

LAMA - GINØ file number of temporary eigenvalue storage file - integer - input.

PHID - GINØ file number of temporary eigenvector storage file - integer - input.

NFØUND - Number of eigenvalues found - integer - output.

SCR1,SCR2, ...,SCR8 - GINØ file numbers of 8 scratch files - integer - input.

4. Method: The overall flow and theoretical considerations of the Determinant Method are explained in section 4.88. Two refinements are made in CDETM. The first is the handling of multiple search regions, which allows the user to control the distribution of starting points in the complex plane. See the EIGC bulk data card description in section 2 of the User's Manual for further details. The second is the use of the EIGP card tc define poles which will be swept from the determinant as if they were previously accepted eigenvalues. This allows the user to prevent convergence to known or already extracted eigenvalues.

5. Design Requirements: CDETM requires two complex double precision d set vectors plus one GINØ buffer in core.

4.59.8.15 Subroutine Name: CDETM2

1. Entry Point: CDETM2

2. Purpose: To arrange 3 starting points in order of the magnitude of the determinant.

3. Calling Sequence: CALL CDETM2(P,D,IP,PR,PI,DR,DI,IPS)

P - Three starting point values - input-complex double precision.

D - Scaled determinants at P - input-complex double precision.

IP - Scale factors for D - input - integer.

PR - Real parts of the reordered starting points - output-double precision.

PI - Imaginary parts of the reordered starting points - output-double precision.

DR - Real parts of the reordered determinants - output-double precision.

DI - Imaginary parts of the reordered determinants - output-double precision.

IPS - Scale factors of the reordered determinants - output - integer.

4.59.8.16 Subroutine Name: CSUMM

1. Entry Point: CSUMM

2. Purpose: To add two scaled complex numbers together.

3. Calling Sequence: CALL CSUMM (D1,D2,ID1,D3,D4,ID2,D5,D6,ID3)

The arguments are defined in the following equation:

$$(D1,D2) \times 10^{ID1} + (D3,D4) \times 10^{ID2} = (D5,D6) \times 10^{ID3}, \tag{18}$$

where all Di's are double precision.

4.59.8.17 Subroutine Name: CSQRT

1. Entry Point: CSQRT

2. Purpose: To compute the positive principal square root of a scaled complex number.

3. Calling Sequence: CALL CSQRT (D1,D2,ID1,D3,D4,ID2)

The arguments are defined in the following equation:

$$(D3,D4) \times 10^{ID2} = \sqrt{(D1,D2) \times 10^{ID1}} \tag{19}$$

where all Di's are double precision.

4.59.8.18 Subroutine Name: CDTFBS

1. Entry Point: CDTFBS

2. Purpose: To solve for the eigenvector given the decomposed incedance matrix.

3. Calling Sequence: CALL CDTFBS (F,EV,BUFFER(1),FU,NRØW)

F - Applied complex load vector - input-complex double precision.

EV - Eigenvector - output- complex double precision.

BUFFER(1) - GINØ buffer.

FU - Matrix control block for [U] - integer - input.

NRØW - Order of problem - integer - input.

4.59.8.19 Subroutine Name: CDETM3

1. Entry Point: CDETM3

2. Purpose: To rescale a scaled complex number.

3. Calling Sequence: CALL CDETM3(D1,D2,ID1)

Let $\overline{D1}$, $\overline{D2}$, $\overline{ID1}$ be the input values of D1, D2, ID1. On return from CDETM3

$$(D1,D2) \times 10^{ID1} = (\overline{D1},\overline{D2}) \times 10^{\overline{ID1}} , \tag{20}$$

and

$$1.0 \leq |(D1,D2)| \leq 10.0 , \tag{21}$$

where all Di's are double precision.

4.59.9 Design Requirements

Open core is defined at /CEAD1X/ to process EED. Open core is defined at /CEADA1/ for use by CEAD1A.

4.60 FUNCTIONAL MODULE VDR (VECTOR DATA RECOVERY)

4.60.1 Entry Point: VDR

4.60.2 Purpose

VDR formats data blocks for input to the Output File Processor (ØFP) and XY plot (XYPLØT) modules to provide a capability for output of vectors in the solution set.

4.60.3 DMAP Calling Sequence

```
VDR     CASECC,EQDYN,USETD,UDV,PP,XYCDB,PNL/ØUDV1,ØPNL1/
        C,N,{TRANRESP / FREQRESP / CEIGN} /C,N,{DIRECT / MØDAL} /V,N,SØRT2/V,N,ØUTPUT/V,N,SDR2/V,N,FMØDE   $
```

4.60.4 Input Data Blocks

CASECC - Case Control Data Table.

EQDYN - Equivalence between external and internal number - Dynamics.

USETD - Displacement set definitions table - Dynamics.

UDV - Partition of Displacement Vector.

PP - Dynamic Load Vector.

XYCDB - XY Control Data Block.

PNL - Non-Linear Load Vector.

Notes:

1. CASECC, EQDYN and USETD may not be purged.
2. PP may be purged only if UDV is purged.
3. PNL and XYCDB may be purged.

4.60.5 Output Data Blocks

ØUDV1 - Output Displacement Requests - Solution set.

ØPNL1 - Output Non-Linear Load Requests - Solution set.

Note: Output data blocks may be purged.

4.60.6 Parameters

The first parameter indicates a Rigid Format and must be one of the three names shown above. The second parameter indicates a direct or modal formulation and must be one of the two names shown above.

SØRT2 - Output-integer-no default. +1 if any SØRT2 output is requested, -1 otherwise.

ØUTPUT - Output-integer-no default. +1 if any output in the solution set is requested, -1 otherwise.

SDR2 - Output-integer-no default. +1 if any requests for output in the physical set are found in CASECC or XYCDB, -1 otherwise.

FMØDE - Input-integer-no default. If a modal formulation, FMØDE = mode number of the first mode. FMØDE is not used in a direct formulation.

4.60.7 Method

4.60.7.1 General

VDR is the main control program for the module. VDRA is called to analyze the Case Control (CASECC) and XYCDB data blocks. If any requests for solution set output are found, VDRB is called to assemble the ØUDV1 output data block for processing by the ØFP. If the problem is a transient response problem, VDRB is called a second time to process any requests for non-linear load output.

4.60.7.2 Analysis of the Case Control and XYCDB Data Blocks

VDRA attempts to open the XYCDB data block. If it is purged, a return is given to VDR. Otherwise, the header record and first data record of XYCDB are skipped, and data applying to all subcases are read from the second data record. If no such data exist, a dummy master is created. Otherwise, the master data are reduced to a list of unique pairs. If only master data exist, flags are set appropriately.

For each record in the Case Control data block the following processing occurs:

1. The record is read into core. If no XYCDB subcase corresponds to the Case Control subcase, pointers are set to the master data. Otherwise, the master data and appropriate XYCDB subcase data are merged and reduced to unique pairs.

2. For each request for solution set output in XYCDB, the corresponding request in Case Control is examined. If no request is present in Case Control, the XYCDB request is reduced to a set in Case Control format, and a request for the set is turned on in Case Control. If the Case Control set is "ALL", no further action is taken. If the Case Control request is a set, the set is merged with the XYCDB set, and the request altered to reflect the new set (unless all points in the XYCDB set were already in the Case Control set). A flag is set if any new requests are formed.

3. When all requests for the current Case Control record have been analyzed, the record (as modified) is written on a scratch file.

4. When all Case Control records have been read, the GINØ file name for the Case Control data block is switched to the scratch file (unless no modifications were made to Case Control).

4.60.7.3 Preparation of Solution Set Output

The operations of VDRB are dependent on the Rigid Format being executed. VDRB operates in all six of the dynamics Rigid Formats. The initial operations in VDRB proceed as follows:

1. For a direct solution, or a modal solution with extra points, the second record of EQDYN is read into core. USETD is read into core.

2. If the problem is a direct solution, each entry in EQDYN is processed. The scalar index value (the 2nd word of each entry) is replaced by the scalar index value in the solution set plus a code indicating which components of the point are in the solution set.

3. If the problem is a modal solution with extra points, the scalar index of each extra point in EQDYN is replaced with a scalar index in the solution set. The scalar indices of all other points are replaced with zero.

4. If the problem is a complex eigenvalue problem, a list of mode numbers and complex eigenvalues is read into core from the CLAMA data block.

5. If the problem is a transient response problem, a list of times is read into core from the PP data block.

6. If the problem is a frequency response problem, a list of frequencies is read into core from the PP data block.

7. The header record on the input file is skipped, and various parameters are initialized for the overall processing.

A record on the Case Control data block is read. The output request is examined. If the output is defined in terms of a set, pointers to the set definition are computed. The vector is unpacked in core (unless the vector is already in core in the case of velocities and accelerations for frequency problems).

Information is assembled to write the identification record on the output data block as follows.

1. For complex eigenvalues, the mode number and eigenvalue are picked up from the list in core.

2. For frequency response, the frequency is picked up from the list in core. A comparison with the ØFREQ selection in Case Control is made. If the current frequency is not marked for output, the remainder of the calculations for the current vector are skipped.

3. For a transient problem, the time is picked up.

The identification record is written. Entries are written in the data record according to the request. The modified EQDYN table in core is used to pick up points in the vector to be output. Conversion to magnitude and phase is made if requested.

When all points in the current request have been processed, post processing occurs depending on the problem type as follows:

1. For complex eigenvalues, a pointer is updated to the next mode number and eigenvalue. If all eigenvectors have not been processed, the steps above are repeated. Otherwise, terminal processing is initiated.

2. For frequency response, if the vector just processed was a displacement vector, the corresponding velocity vector is determined by differentiating with respect to time.

$$\{v\} = iw\ \{u\}. \tag{1}$$

Similarly, if the vector just processed was a velocity vector, the corresponding acceleration vector is formed by differentiating with respect to time:

$$\{a\} = iw\ \{v\}. \tag{2}$$

If all vectors have not been processed, the steps above are repeated. Otherwise, terminal processing is initiated.

3. For transient response, pointers are updated so that the vectors will be processed in the order a) displacement, b) velocity, and c) acceleration. If all vectors have not been processed, the steps above are repeated. Otherwise, terminal processing is initiated.

The terminal processing consists of closing all files, writing a trailer on the output file and exiting.

4.60.8 Subroutines

4.60.8.1 Subroutine Name: VDR

1. Entry Point: VDR
2. Purpose: Main control program for the module.
3. Calling Sequence: CALL VDR

4.60.8.2 Subroutine Name: VDRA

1. Entry Point: VDRA
2. Purpose: To analyze the output requests in the Case Control and XYCDB data blocks.
3. Calling Sequence: CALL VDRA

4.60.8.3 Subroutine Name: VDRB

1. Entry Point: VDRB

2. Purpose: Tc process requests for solution set output and assemble the output data block.

3. Calling Sequence: CALL VDRB (INFIL,ØUTFL,IREQ)

INFIL - GINØ file name of the data block containing vectors to be output in the solution set.

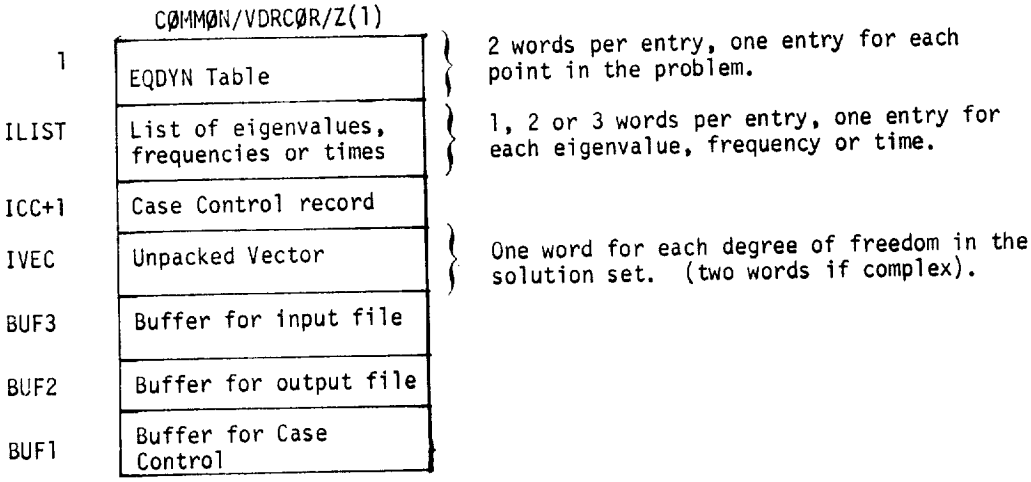
ØUTFL - GINØ file name of the data block where solution set output will be written.

IREQ - Word position in the Case Control record where solution set output request is defined.

4.60.9 Design Requirements

4.60.9.1 Allocation of Core Storage

The maximum storage requirements for the module are in VDRB. A general picture of core storage is as follows:



4.60.9.2 Environment

The Block Data program VDRBD initializes /VDRCØM/ with GINØ file names, data defining position of parameters in a Case Control record, data defining rigid formats and problem types, and miscellaneous data. It must be in core when VDR is executed.

The module VDR is designed to be executed as one overlay segment. Open core is defined by /VDRCØR/. Two scratch files are used.

4.61 FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH)

4.61.1 Entry Point: FRRD

4.61.2 Purpose

To solve the matrix equation

$$[-\omega_i^2 [M] + i\omega [B] + [K]] [X] = [P(\omega_i)] \quad (1)$$

at a given set of frequencies ω_i and loads P (which may be functions of ω_i).

4.61.3 DMAP Calling Sequence

```
FRRD    CASECC,USETD,DLT,FRL,GMD,GØD,KHH,BHH,MHH,PHIDH,DIT/UHV,PS,PD,PP/V,N,APP/V,N,FØRM/
        V,N,LUSETD/V,N,MPCF1/V,N,SINGLE/V,N,ØMIT/V,N,NØNCUP/V,N,FRQSET    $
```

4.61.4 Input Data Blocks

CASECC - Case Control Data table.

USETD - Displacement set definitions table dynamics.

DLT - Dynamic Loads Table.

FRL - Frequency Response List.

GMD - Multipoint constraint transformation matrix - dynamics.

GØD - Omitted coordinate transformation matrix - dynamics.

KHH - Modal stiffness matrix - h set.

BHH - Modal damping matrix - h set.

MHH - Modal mass matrix - h set.

PHIDH - Transformation matrix from d set to modal coordinates.

DIT - Direct Input Tables.

Notes:
1. CASECC cannot be purged.
2. USETD cannot be purged.
3. DLT cannot be purged.
4. FRL cannot be purged.
5. GMD cannot be purged if MPCF1 $\geq$ 0.
6. GØD cannot be purged if ØMIT $\geq$ 0.

7. PHIDH cannot be purged if FØRM = MØDAL.
8. DIT cannot be purged if a load uses tables.

4.61.5 Output Data Blocks

UHV - Displacement vectors.

PS - Partition of load vector matrix giving loads in s set.

PD - Load vectors - d set.

PP - Load vectors - p set.

Notes: 1. UHV, PD, and PP cannot be purged.
2. PS cannot be purged if SINGLE $\geq$ 0.

4.61.6 Parameters

APP - Input-BCD-no default. APP should be set equal to DISP.

FØRM - Input-BCD-no default. FØRM = MØDAL implies a modal solution should be used.

LUSETD - Input-integer-no default. LUSETD indicates length of p set.

MPCF1 - Input-integer-no default. MPCF1 $\geq$ 0 implies multipoint constraints present.

SINGLE - Input-integer-no default. SINGLE $\geq$ 0 implies single-point constraints present.

ØMIT - Input-integer-no default. ØMIT $\geq$ 0 implies omitted coordinates present.

NØNCUP - Input-integer-no default. NØNCUP= -1 implies noncoupled solution if FØRM = MØDAL.

FRQSET - Output-integer-no default. FRQSET is the set id of the selected frequency list from CASECC.

4.61.7 Method

4.61.7.1 Overview of the Method

The Frequency Response module for the displacement approach assembles a frequency-dependent load vector and solves for the steady-state, frequency response, displacement vectors. Various load sets are defined as functions of frequency. Combinations of these sets are used with the various specified frequencies. Load vectors for each frequency are formed and reduced to loads on the proper degree of freedom. The solutions for both direct formulation and coupled modal formulation are identical except that different matrices are used. The solution involves a

triangular decomposition and back substitution using an unsymmetric, complex matrix and complex load vectors for each frequency. The solutions for the uncoupled modal formulation are analytic equations.

4.61.7.2 Logical Phases

1. The load vectors for each desired frequency are assembled from the DLT data block. The DLØAD section of the DLT tells which load sets to use and what scale factors to use in combining the load sets. The data for each load set are given in the RLØAD section of the DLT. This work is done in subroutine FRRD1A.

2. The total load vectors are partitioned and manipulated to produce load vectors on the solution coordinates. This work is done in subroutine FRRD1B.

3. The matrix equation for displacements is now solved for each load combination and each frequency. The overall dynamic matrix is formed. The matrix is decomposed, and the displacements are formed by back substitution using the various loads. If the formulation is an uncoupled modal system, the displacements are calculated directly. This work is done by subroutines FRRD1C and FRRD1D or FRRD1F.

4. The solution vectors are then resorted into load-frequency order. This work is done by subroutine FRRD1E.

4.61.7.3 Algorithms

1. Assembly of Load Vectors:

The frequency set id is extracted from CASECC. This frequency set is placed in core from the FRL and converted from radians to frequency. These frequencies are output into the header of PP for later output identification. The load id is read from CASECC, found in DLT, and a table is constructed giving a simple id and a scale factor for each component. The DLT data are read for each simple id, and a list of the required tables is extracted. Core is allocated to hold as many load vectors as possible up to the number of frequencies. If tables are present, they are initialized and evaluated for all frequencies in core. The DLT is read, and two types of loads are constructed:

1) RLØAD1

$$P(f) = A\,[C(f) + iD(f)]\,e^{i(\Theta - 2\pi f\tau)}, \qquad (2)$$

2) RLØAD2

$$P(f) = A\ B(f)\ e^{i(\phi(f) + \Theta - 2\pi f\tau)} \tag{3}$$

where A, B, C, D, Θ, τ and ϕ are user input constants or tables.

If all frequencies cannot be evaluated at once, additional passes through the DLT are made until all are evaluated. If additional subcases exist in CASECC, the above steps are repeated for each load.

2. Manipulation of Load Vectors:

The vectors produced in the previous sections are related to the p set. They are reduced by the following steps using data blocks USETD, GMD and GØD.

If MPCF1 $\geq$ 0:

$$\{P_p\} \Longrightarrow \begin{Bmatrix} \bar{P}_{ne} \\ P_m \end{Bmatrix}, \tag{4}$$

$$\{P_{ne}\} = \{\bar{P}_{ne}\} + [G_m^d]^T \{P_m\}\ . \tag{5}$$

If SINGLE $\geq$ 0:

$$\{P_{ne}\} \Longrightarrow \begin{Bmatrix} P_{fe} \\ P_s \end{Bmatrix}. \tag{6}$$

$\{P_s\}$ is output on data block PS.

If ØMIT $\geq$ 0:

$$\{P_{fe}\} \Longrightarrow \begin{Bmatrix} \bar{P}_d \\ P_o \end{Bmatrix}, \tag{7}$$

$$\{P_d\} = \{\bar{P}_d\} + [G_o^d]^T \{P_o\}, \tag{8}$$

$\{P_d\}$ is output on PD.

If FØRM = MØDAL:

$$\{P_h\} = [\Phi_{dh}]^T \{P_d\}. \tag{9}$$

3. Solution Phase:

For a direct formulation the equation to be solved is:

$$[-\omega^2 [M_{dd}] + i\omega [B_{dd}] + [K_{dd}]]\ \{u_d\} = \{P_d(\omega)\}\ . \tag{10}$$

For a coupled modal formulation the equation to be solved is:

$$[-\omega^2 [M_{hh}] + i\omega[B_{hh}] + [K_{hh}]] \{u_h\} = \{P_n(\omega)\}. \quad (11)$$

The left hand matrix is generated by two calls to ADD and decomposed. The normal matrix decomposition checks are relaxed in these solutions. It is expected that the matrices will not pass the triangular decomposition at certain frequencies. The solution will proceed, and only a warning will be issued. The loads at the given frequency are collected from the load file and fed to GFBS for a forward backward substitution solution. If the decomposition failed, a zero vector will result.

For one uncoupled modal formulation the equations to be solved are:

$$\{\epsilon_i\} = \left\{ \frac{P_i(\omega)}{-m_i\omega^2 + ib_i\omega + k_i} \right\}. \quad (12)$$

With zero damping the uncoupled modal formulation may produce division by small numbers. This fact is noted and the solution proceeds.

4. Order Phase:

Except for the uncoupled modal approach it may be necessary to reorder the solutions from a frequency / load sort to a load / frequency sort.

4.61.8 Subroutines

Utility subroutines PRETAB,TAB,CALCV,SSG2B,SSG2A,SSG2C,CDCØMP, and GFBS are used. See subroutine descriptions, section 3 for details.

4.61.8.1 Subroutine Name: FRRD1A

1. Entry Point: FRRD1A

2. Purpose: To assemble the user selected loads.

3. Calling Sequence: CALL FRRD1A (DLT,FRL,CASECC,DIT,PP,LUSETD,NFREQ,NLØAD,FRQSET)
DLT,FRL,CASECC,DIT,PP are GINØ file numbers of their respective data blocks - integer - input.

LUSETD - Length of p set - integer - input.

NFREQ - Number of frequencies in selected frequency set - integer - output.

NLØAD - Number of loads (records in CASECC) selected - integer - output.

FRQSET - Set id of selected frequency set - integer - output.

4.61.8.2 Subroutine Name: FRRD1B

1. Entry Point: FRRD1B

2. Purpose: To reduce loads from the p to the d (or h) set.

3. Calling Sequence: CALL FRRD1B (PP,USETD,GMD,GØD,MULTI,SINGLE,ØMIT,MØDAL,PHIDH, PD,PS,PH,SCR1,SCR2,SCR3,SCR4)

PP,USETD,GMD,GØD,PHIDH,PD,PS,PH are GINØ file numbers of their respective data blocks - integer - input.

MULTI - MULTI $\geq$ 0 implies m's are present - integer - input.

SINGLE - SINGLE $\geq$ 0 implies s's are present - integer - input.

ØMIT - ØMIT $\geq$ 0 implies o's are present - integer - input.

MØDAL - MØDAL = MØDA implies a modal formulation - BCD - input.

SCR1,...,SCR4 - GINØ file numbers of 4 scratch files - integer - input.

4.61.8.3 Subroutine Name: FRRD1C

1. Entry Point: FRRD1C

2. Purpose: To form and decompose "left" hand side of the frequency equation.

3. Calling Sequence: CALL FRRD1C (FRL,FRQSET,MDD,BDD,KDD,I,ULL,LLL,SCR1,SCR2,SCR3, SCR4,IGØØD)

FRL,MDD,BDD,KDD,ULL,LLL,SCR1-4 are GINØ file numbers of their respective data blocks - integer - input.

FRQSET - Set id of selected frequency set - integer - output.

I - Current frequency counter - integer - input.

IGØØD - IGØØD = 1 implies a singular matrix - integer - output.

4.61.8.4 Subroutine Name: FRRD1D

1. Entry Point: FRRD1D

2. Purpose: To solve for displacements given decomposition factors and loads.

3. Calling Sequence: CALL FRRD1D (PD,ULL,LLL,SCR1,SCR2,UDVP,I,NLØAD,IGØØD,NFREQ)

PD,ULL,LLL,UDVP,SCR1,SCR2 are GINØ file numbers of their respective data blocks - integer - input.

I - Current frequency count - integer - input.

NLØAD - Number of loads - integer - input.

IGØØD - IGØØD = 1 implies a singular matrix - integer - input.

NFREQ - Total number of frequencies - integer - input.

4.61.8.5 Subroutine Name: FRRD1E

1. Entry Point: FRRD1E

2. Purpose: To reorder displacements if necessary.

3. Calling Sequence: CALL FRRD1E (UDVP,UDV,NLØAD,I)

UDVP - GINØ file number of displacements sorted by frequency/load - integer - input.

UDV - GINØ file number of displacements sorted by load/frequency - integer - input.

NLØAD - Number of loads - integer - input.

I - Number of frequencies solved.

4.61.8.6 Subroutine Name: FRRD1F

1. Entry Point: FRRD1F

2. Purpose: To solve the uncoupled modal equations.

3. Calling Sequence: CALL FRRD1F (MHH,BHH,KHH,FRL,FRQSET,NLØAD,NFREQ,PH,UHV)

MHH,BHH,KHH,FRL,PH,UHV are GINØ file numbers of their respective data blocks. - integer - input.

FRQSET - Selected frequency set id-integer - input.

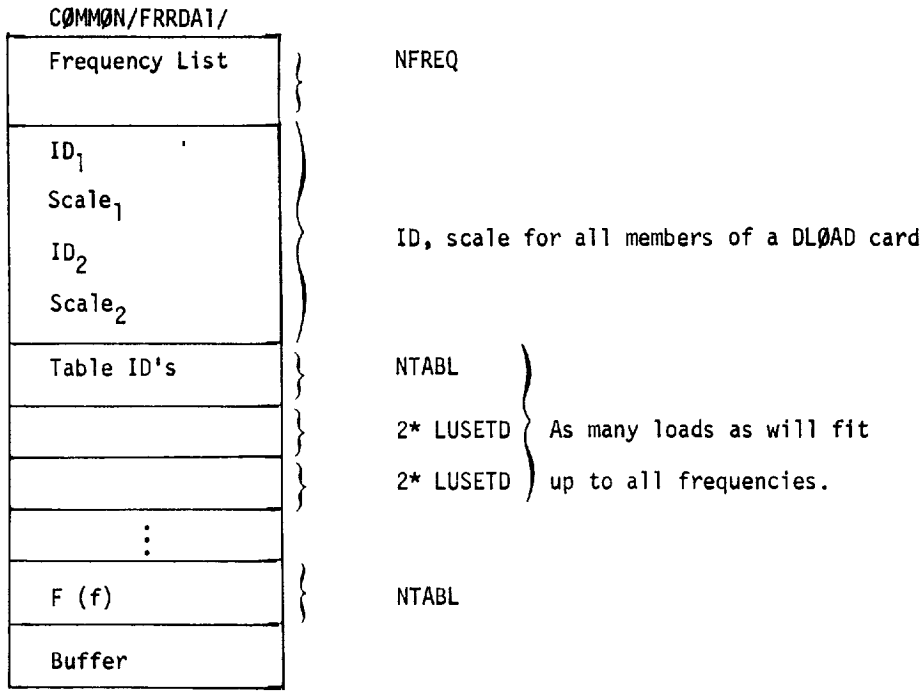
NFREQ - Number of frequencies in FRQSET - integer - input.

NLØAD - Number of loads (Subcases in current execution) - integer - input.

4.61.9 Design Requirements

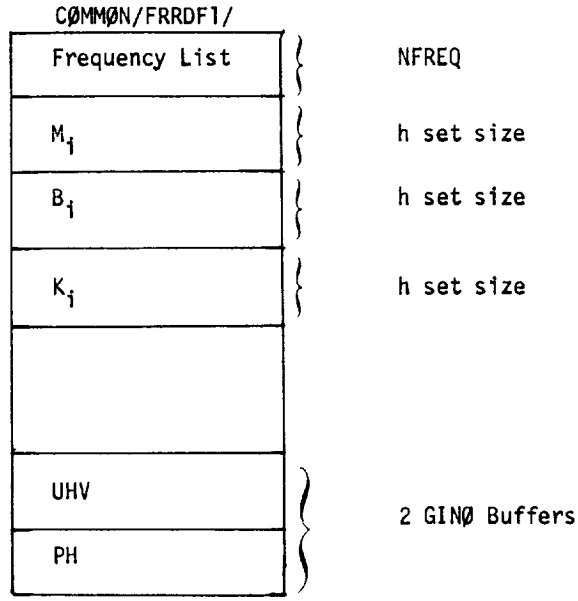
Eight scratch files are used by FRRD.

Open core at /FRRDA1/ is used as follows:



Open core at /FRRDB1/, /FRRDC1/, /FRRDD1/ are used by the matrix routines.

Open core at /FRRDF1/ is used as follows:



4.61.10 Diagnostic Messages

Module FRRD may issue the following diagnostic messages:

3005, 3008 and 3045.

4.62 FUNCTIONAL MODULE SDR3 (STRESS DATA RECOVERY - PHASE 3 - SØRT1 to SØRT2 PROCESSOR)

4.62.1 Entry Point: SDR3

4.62.2 Purpose

To transpose (perform SØRT2) data blocks containing data prepared for output in the form of ELEMENT-ID-SETS or PØINT-ID-SETS versus TIME-STEP or FREQUENCY-STEP to data prepared for output in the form of TIME-STEP-SETS or FREQUENCY-STEP-SETS versus ELEMENT-ID or PØINT-ID.

4.62.2.1 Example of SØRT1 and SØRT2 Output

Below is a table of ØFP printed output of SDR3 input (SØRT1) and output (SØRT2) data blocks.

SDR3 Input Data Block Printed (SØRT1)

TIME = 1.0			D I S P L A C E M E N T S			
POINT-ID	T1	T2	T3	R1	R2	R3
1	0.0	4.53	0.0	0.0	0.0	0.0
2	0.0	5.12	0.0	0.0	0.0	0.0
TIME = 2.0			D I S P L A C E M E N T S			
POINT-ID	T1	T2	T3	R1	R2	R3
1	0.0	4.83	0.0	0.0	0.0	0.0
2	0.0	5.53	0.0	0.0	0.0	0.0
TIME = 3.0			D I S P L A C E M E N T S			
POINT-ID	T1	T2	T3	R1	R2	R3
1	0.0	6.84	0.0	0.0	0.0	0.0
2	0.0	7.96	0.0	0.0	0.0	0.0

SDR3 Output Data Block Printed (SØRT2)

POINT-ID = 1			D I S P L A C E M E N T S			
TIME	T1	T2	T3	R1	R2	R3
1.0	0.0	4.53	0.0	0.0	0.0	0.0
2.0	0.0	4.83	0.0	0.0	0.0	0.0
3.0	0.0	6.84	0.0	0.0	0.0	0.0
POINT-ID = 2			D I S P L A C E M E N T S			
TIME	T1	T2	T3	R1	R2	R3
1.0	0.0	5.12	0.0	0.0	0.0	0.0
2.0	0.0	5.53	0.0	0.0	0.0	0.0
3.0	0.0	7.96	0.0	0.0	0.0	0.0

4.62.3 DMAP Calling Sequence

SDR3 IN1,IN2,IN3,IN4,IN5,IN6/ØUT1,ØUT2,ØUT3,ØUT4,ØUT5,ØUT6/ $

4.62.4 Input Data Blocks

One to six data blocks in any order desired. Input data blocks to SDR3 which are purged are ignored.

4.62.5 Output Data Blocks

One to six data blocks in corresponding order to that of the input data blocks. If SØRT2 is to be performed, there must be an available output data block for the corresponding input data block (Non-Fatal Error if this condition is not met).

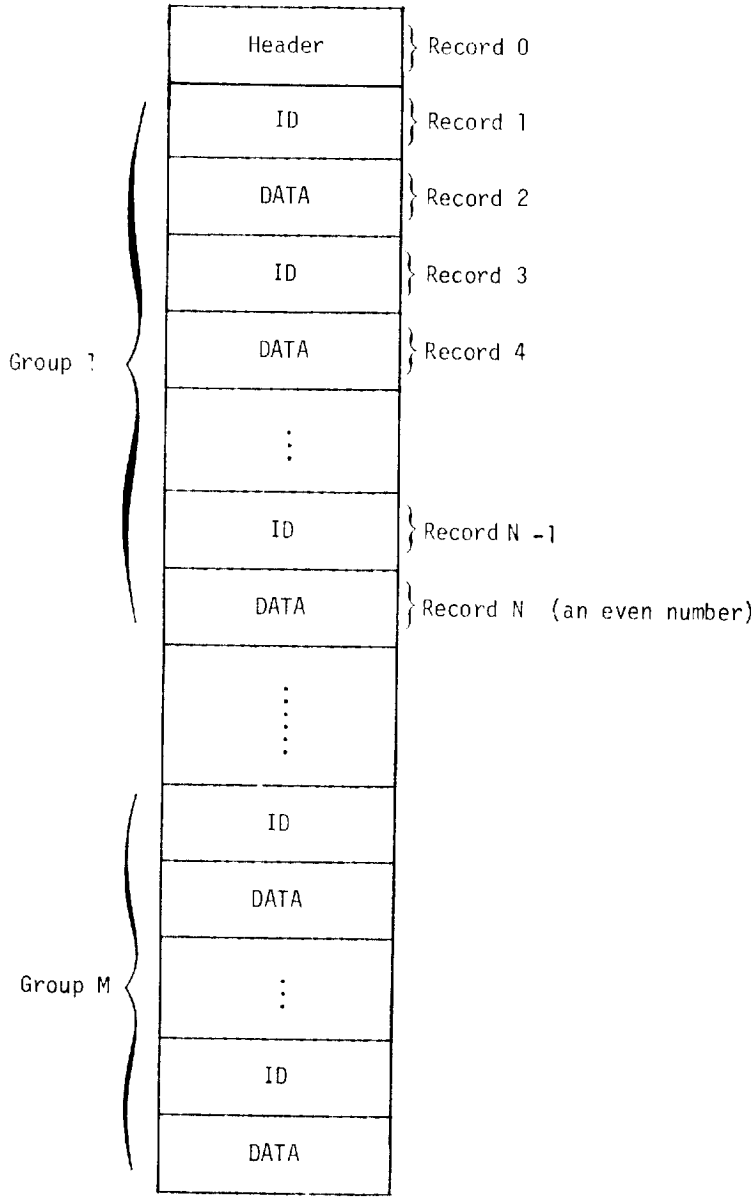
4.62.6 Parameters

None

4.62.7 Method

4.62.7.1 Input and Output Data Block Record Arrangements

Both the input and output data blocks of SDR3 have the following format:

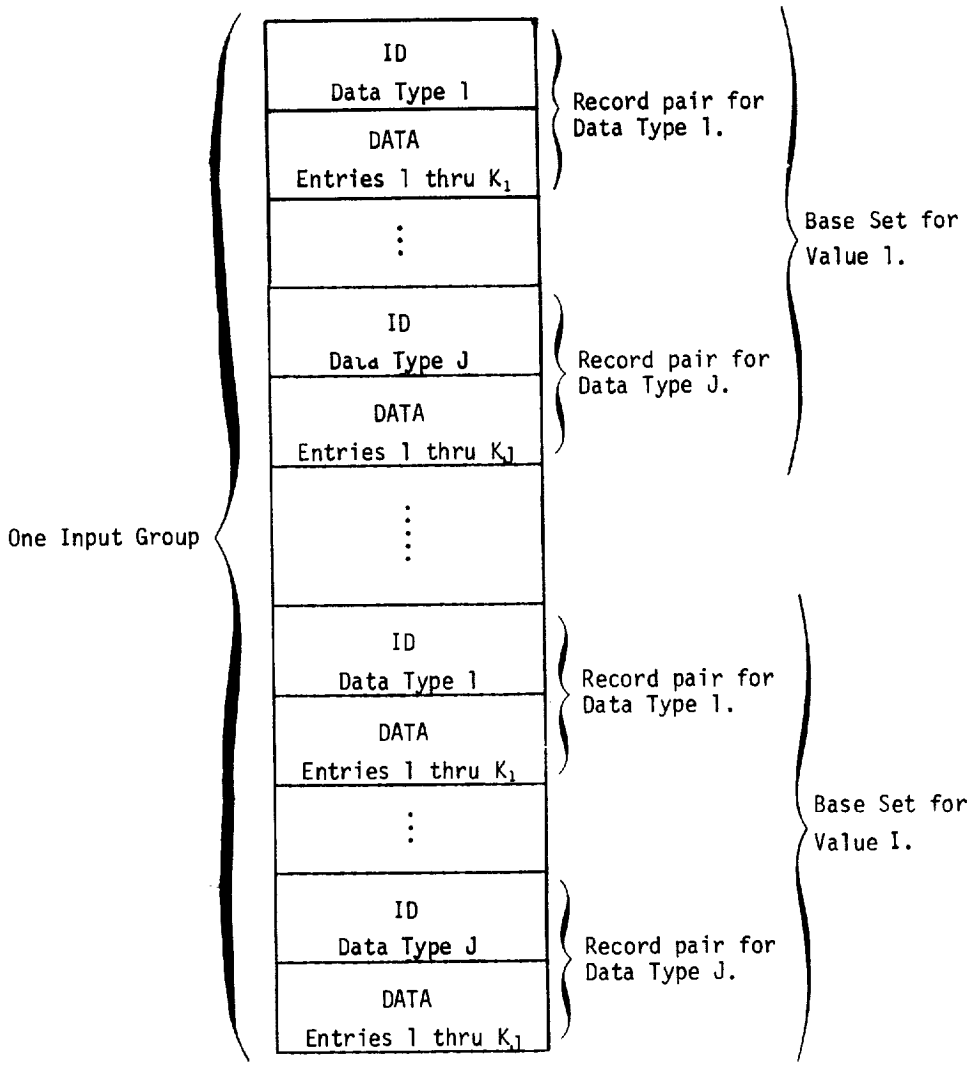

Header
Record 0
ID
Record 1
DATA
Record 2
ID
Record 3
DATA
Record 4
Group 1
ID
Record N -1
DATA
Record N (an even number)
ID
DATA
Group M
ID
DATA

4.62.7.2 Description of a Group

1. An input (SØRT1) data block Group and an output (SØRT2) data block Group are given in the following figures:

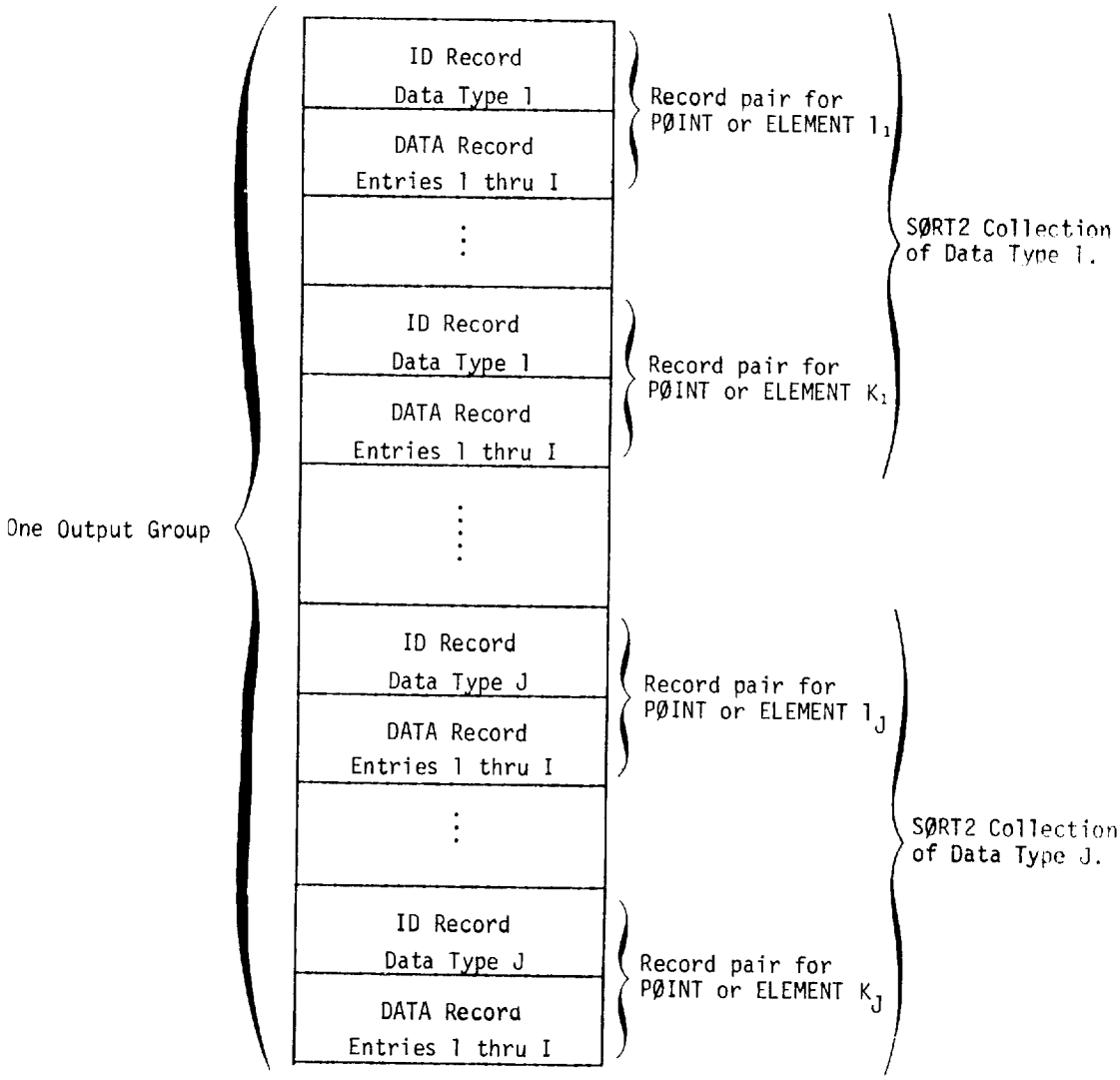
Theoretical Input Group

(SØRT1)



Theoretical Output Group

(SØRT2)



2. In the above figures each Group is independent of any other Group so far as SDR3 need be concerned.

3. A Group is defined as a collection of successive records belonging to the same subcase.

4. An ID-Record is of a fixed size equal to 146 words.

5. A DATA-Record contains multiple Entries with each Entry being of a length in words specified within the immediately preceding ID-Record.

6. I = The number of Values (FREQUENCIES or TIMES) present in the Group.

7. A Base Set is a sub-Group of the Group containing data records for one particular Value.

8. J = The number of different Data Types (DISPLACEMENTS, VELOCITIES, etc.) within a Base Set.

9. K_j = The number of Entries for Data Type j.

10. Respective records of any two Base Sets within an input data block Group are of the same size.

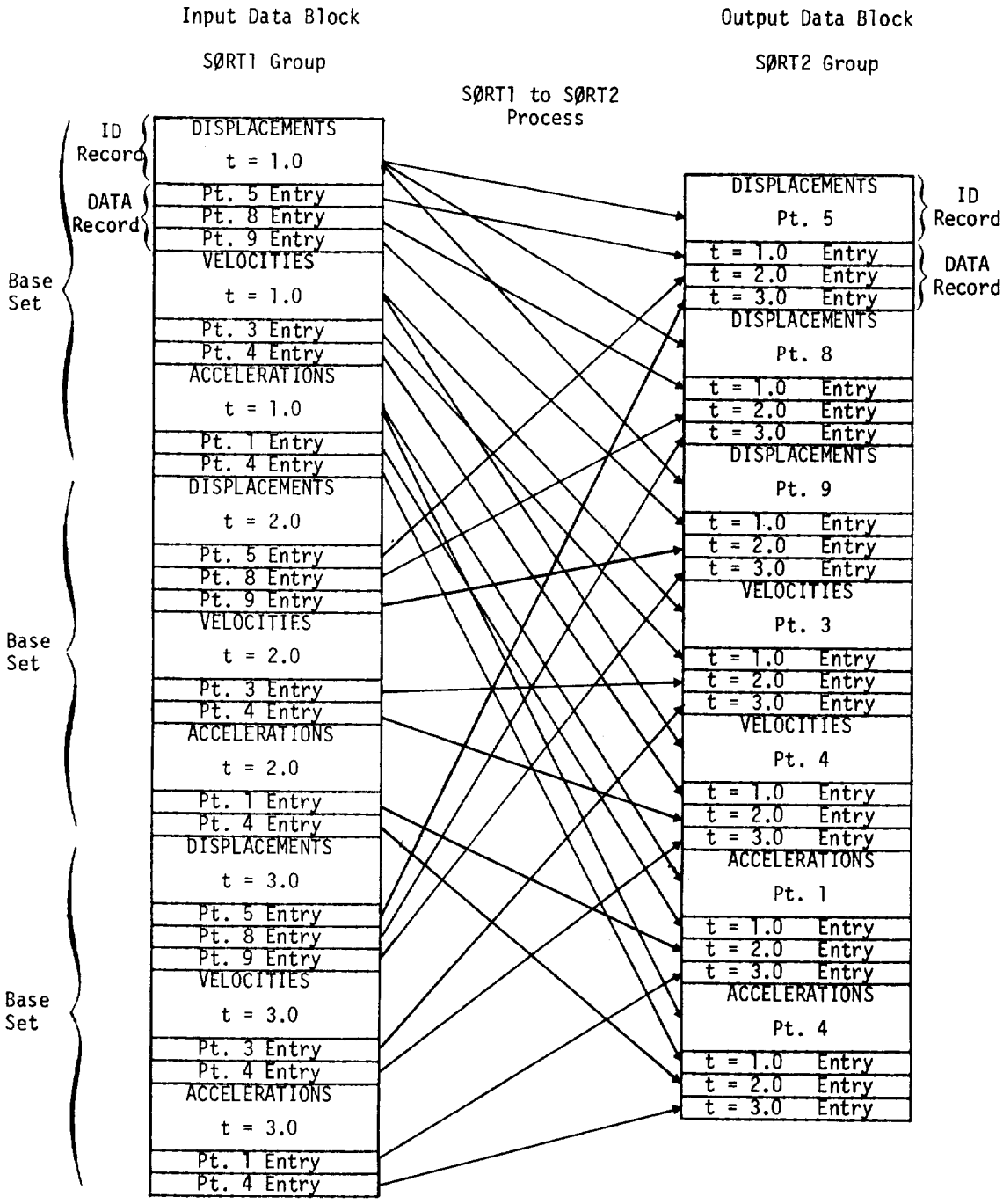
11. Respective Entries within respective DATA Records of all Base Sets of an input data block Group begin with the same ELEMENT-ID or PØINT-ID.

12. Most input data blocks will contain only one Group having but one Data Type. There is normally more than one Base Set within any Group.

13. A pictoral representation of a SØRT1 to SØRT2 process is given on the next page using the following data:

Values = 3 time steps (1.0, 2.0, 3.0)

Data Types = {1 - Displacements (3 Entries/Value - points 5, 8 and 9)
2 - Velocities (2 Entries/Value - points 3 and 4)
3 - Accelerations (2 Entries/Value - points 1 and 4)


Input Data Block
SØRT1 Group
SØRT1 to SØRT2 Process
Output Data Block
SØRT2 Group
ID Record
DATA Record
Base Set
DISPLACEMENTS t = 1.0
Pt. 5 Entry
Pt. 8 Entry
Pt. 9 Entry
VELOCITIES t = 1.0
Pt. 3 Entry
Pt. 4 Entry
ACCELERATIONS t = 1.0
Pt. 1 Entry
Pt. 4 Entry
DISPLACEMENTS t = 2.0
Pt. 5 Entry
Pt. 8 Entry
Pt. 9 Entry
VELOCITIES t = 2.0
Pt. 3 Entry
Pt. 4 Entry
ACCELERATIONS t = 2.0
Pt. 1 Entry
Pt. 4 Entry
DISPLACEMENTS t = 3.0
Pt. 5 Entry
Pt. 8 Entry
Pt. 9 Entry
VELOCITIES t = 3.0
Pt. 3 Entry
Pt. 4 Entry
ACCELERATIONS t = 3.0
Pt. 1 Entry
Pt. 4 Entry
DISPLACEMENTS Pt. 5
t = 1.0 Entry
t = 2.0 Entry
t = 3.0 Entry
DISPLACEMENTS Pt. 8
t = 1.0 Entry
t = 2.0 Entry
t = 3.0 Entry
DISPLACEMENTS Pt. 9
t = 1.0 Entry
t = 2.0 Entry
t = 3.0 Entry
VELOCITIES Pt. 3
t = 1.0 Entry
t = 2.0 Entry
t = 3.0 Entry
VELOCITIES Pt. 4
t = 1.0 Entry
t = 2.0 Entry
t = 3.0 Entry
ACCELERATIONS Pt. 1
t = 1.0 Entry
t = 2.0 Entry
t = 3.0 Entry
ACCELERATIONS Pt. 4
t = 1.0 Entry
t = 2.0 Entry
t = 3.0 Entry
ID Record
DATA Record

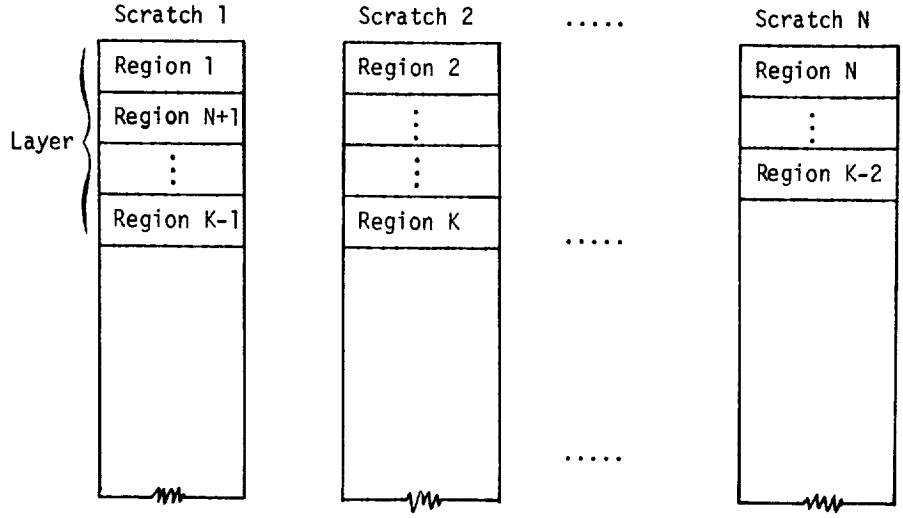
4.62.7.3 Physical Data Processing (SØRT1 to SØRT2)

All emphasis is placed on the Group, and thus in performing SØRT2 a Group pointer always points to the first record of the current Group being processed.

Each Group is processed and completed successively until all Groups have been processed. For each Group a loop of J passes is executed. During the j^{th} pass of this loop, the j^{th} Data Type (note 4.62.7.2) present of the Base Sets will be collected and transposed. The transpose consists of determining how many Entries are present for the current Data Type and then dividing the available core into that many Regions. The Entries of each DATA record for the j^{th} Data Type are distributed in Entry order, one each, to the Regions. At the time each Entry is distributed to a Region, the Entry's first word (PØINT-ID or ELEMENT-ID) is replaced by the Value (FREQUENCY or TIME) in the ID-Record associated with the DATA-Record from which the Entry has come. At the conclusion of each pass of this loop, output to the data block can proceed. For each Region an ID-Record is written. This ID-Record is a copy of the input data block ID-Record in the first Base Set for the j^{th} Data Type, having had the Value (FREQUENCY or TIME) replaced with the PØINT-ID or ELEMENT-ID of the respective Region. The filled portion of the Region is then output as the DATA-Record.

4.62.7.4 Spill Logic

If during the Entry distribution the Regions can hold no more Entries, spill to scratch files is performed. A Layer of records is written, one record for each Region, each time spill is required.



At the output stage, if spill to the scratch files has occurred, the Regions in the scratch files are output before the in-core Regions.

4.62.8 Subroutines

4.62.8.1 Subroutine Name: SDR3A

1. Entry Point: SDR3A

2. Purpose: To perform all SØRT2 operations when called by the driver routine SDR3.

3. Calling Sequence: CALL SDR3A (ØFPFIL)

   ØFPFIL - An array of six words, one for each input data block, each of which is set to zero before the CALL and then reset by SDR3A with a traceback positive integer in the event an error for its respective data block occurred.

4.62.9 Design Requirements

1. The design requires that the largest DATA-Record fit in core. If a problem is outputting so many ELEMENT-ID or PØINT-ID Entries for a particular FREQUENCY or TIME that core is insufficient, then more subcases in conjunction with output request sets are recommended.

2. CØMMØN/SDR3ZZ/Z(1)

This common block defines open core for the SDR3 module.

4.62.10 Diagnostic Messages

All errors within SDR3 are considered non-fatal-User Warning type errors. Any error resulting in termination of the SØRT2 process results in the setting of an SDR3 traceback number, an appropriate message, and a call to the ØFP (Output File Processor) which in turn will output the data block in SØRT1 format. If ØFP is unable to output the data block it in turn will call the TABPRT routine, and the data block will be printed.

4.63 FUNCTIONAL MODULE XYTRAN (XY - OUTPUT DATA TRANSLATOR)

4.63.1 Entry Point: XYTRAN

4.63.2 Purpose

To read the first record of the XYCDB data block (prepared by subroutine IFP1XY of Executive module IFP1); to set xy-output parameters from the serial specifications of this record; to interpret the user curve requests; to locate in the XYTRAN input data blocks (2 thru 6) the data sets containing the requested curve data; to prepare summary and xy-coordinate data for the requested curves and output them to the system output printer and punch units; and to prepare xy-coordinate data and output them to the XYTRAN output data block for direct plotting by the XYPLØT module of those curve requests specified to be plotted.

4.63.3 DMAP Calling Sequences

4.63.3.1 Frequency Response - Direct Formulation. (Rigid Format 8)

1. Vector data recovery output.

```
XYTRAN  XYCDB,ØUDVC2,,,, /XYPLTFA/C,N,FREQ/C,N,DSET/V,N,PFILE/V,N,CARDNØ  $
```

2. Stress data recovery output.

```
XYTRAN  XYCDB,ØPPC2,ØQPC2,ØUPVC2,ØESC2,ØEFC2/XYPLTF/C,N,FREQ/C,N,PSET/V,N,PFILE/V,N,CARDNØ $
```

3. Random response output.

```
XYTRAN   XYCDB,PSDF,AUTØ,,,/XYPLTR/C,N,RAND/C,N,PSET/V,N,PFILE/V,N,CARDNØ  $
```

4.63.3.2 Transient Response - Direct Formulation. (Rigid Format 9)

1. Vector data recovery output.

```
XYTRAN  XYCDB,ØUDV2,ØPNL2,,,/XYPLTTA/C,N,TRAN/C,N,DSET/V,N,PFILE/V,N,CARDNØ  $
```

2. Stress data recovery output.

```
XYTRAN  XYCDB,ØPP2,ØQP2,ØUPV2,ØES2,ØEF2/XYPLTT/C,N,TRAN/C,N,PSET/V,N,PFILE/V,N,CARDNØ  $
```

4.63.3.3 Frequency Response - Modal Formulation. (Rigid Format 11)

1. Vector data recovery output.

```
XYTRAN  XYCDB,ØUHVC2,,,,/XYPLTFA/C,N,FREQ/C,N,HSET/V,N,PFILE/V,N,CARDNØ $
```

2. Stress data recovery output.

```
XYTRAN  XYCDB,ØPPC2,ØQPC2,ØUPVC2,ØESC2,ØEFC2/XYPLTF/C,N,FREQ/C,N,PSET/V,N,PFILE/
V,N,CARDNØ  $
```

3. Random response output.

```
XYTRAN  XYCDB,PSDF,AUTØ,,,/XYPLTR/C,N,RAND/C,N,PSET/V,N,PFILE/V,N,CARDNØ  $
```

4.63.3.4 Transient Response - Modal Formulation. (Rigid Format 12)

1. Vector data recovery output.

```
XYTRAN  XYCDB,ØUHV2,ØPNL2,,,/XYPLTTA/C,N,TRAN/C,N,HSET/V,N,PFILE/V,N,CARDNØ  $
```

2. Stress data recovery output.

```
XYTRAN  XYCDB,ØPP2,ØQP2,ØUPV2,ØES2,ØEF2/XYPLTT/C,N,TRAN/C,N,PSET/V,N,PFILE/V,N,CARDNØ  $
```

4.63.4 Input Data Blocks

XYCDB - XY Output Control Data Block.

ØUDVC2 - Output displacement vector requests (solution set, SØRT2, complex).

ØPPC2 - Output load vector requests (solution set, SØRT2, complex).

ØQPC2 - Output forces of single-point constraint requests (solution set, SØRT2, complex).

ØUPVC2 - Output displacement vector requests (p set, SØRT2, complex).

ØESC2 - Output element stress requests (SØRT2, complex).

ØEFC2 - Output element force requests (SØRT2, complex).

PSDF - Power Spectral Density Table.

AUTØ - Autocorrelation function table.

ØUDV2 - Output displacement vector requests (solution set, SØRT2, real).

ØPNL2 - Output nonlinear load requests (solution set, SØRT2, real).

ØPP2 - Output load vector requests (p set, SØRT2, real).

ØQP2 - Output forces of single-point constraint (p set, SØRT2, real).

ØUPV2 - Output displacement vector requests (p set, SØRT2, real).

ØES2 - Output element stress requests (SØRT2, real).

ØEF2 - Output element force requests (SØRT2, real).

ØUHVC2 - Output displacement vector requests (solution set, SØRT2, complex).

ØUHV2 - Output displacement vector requests (solution set, SØRT2, real).

4.63.5 Output Data Blocks

XYPLTFA -
XYPLTF -
XYPLTR -
XYPLTTA -
XYPLTT -

} XY-Plot output requests prepared by XYTRAN for direct plotting by XYPLØT.

4.63.6 Parameters

CARDNØ - Input and output-integer-default value = 0. CARDNØ is incremented by one and punched in columns 73-80 of each card punched by XYTRAN.

PFILE - Input and output-integer-default value = 0. PFILE is incremented by one for each frame XYTRAN defines for output by XYPLØT.

FREQ - Input-BCD-2-word-constant distinguishes the problem as frequency response.

TRAN - Input-BCD 2-word-constant distinguishes the problem as transient response.

RAND - Input-BCD 2-word-constant distinguishes the problem as random response.

DSET - Input-BCD 2-word-constant distinguishes the input vector as the d set.

PSET - Input-BCD 2-word-constant distinguishes the input vector as the p set.

HSET - Input-BCD 2-word-constant distinguishes the input vector as the h set.

4.63.7 Method

4.63.7.1 The following diagram illustrates the process of serially reading through the XYCDB data block's first record and performing the XYTRAN data processing.

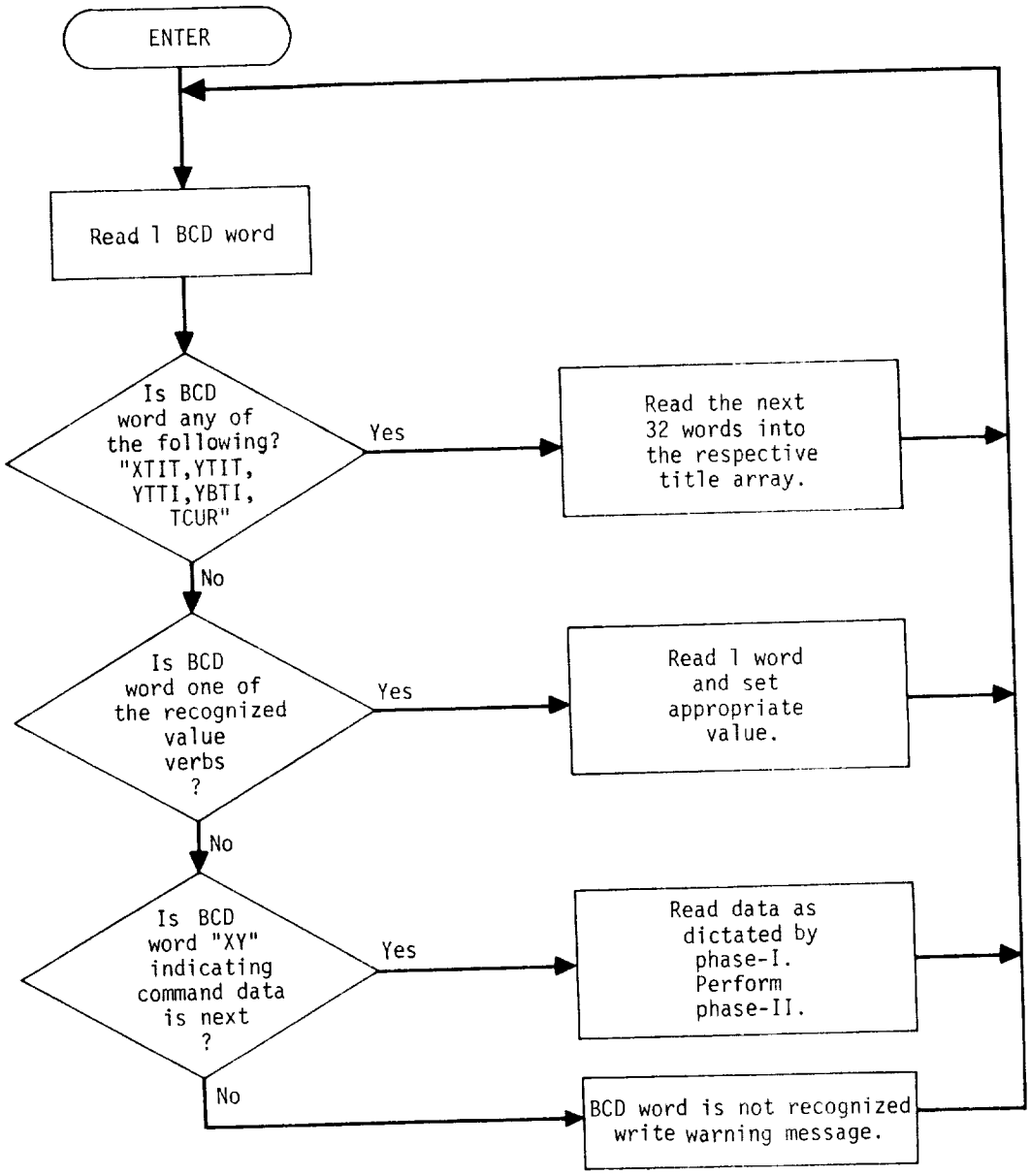


Figure 1. Flowchart for reading the first record of XYCDB

4.63.7.2 Phase I

In Phase I the XYCDB data block is further read to:

1. Determine the type of XY-output curves desired. (Response, Autocorrelation, or Power Spectral Density Function);
2. Determine the type of data (displacements, stresses, etc.,) and subcases desired;
3. Determine which types of XY-output are requested of XYPUNCH, XYPEAK, XYPRINT, and XYPLØT (XY-output requests are described in section 4 of the User's Manual).
4. Determine the point-component curve relationships for a frame.

The data for all curves of a given frame (upper and lower, or whole) are then collected and stored in core.

4.63.7.3 Phase II

The operations of Phase II involve the analysis of the curve data in conjunction with the XY-output specifications stored to this point as a set of values, and the computation and setting of dynamic curve limits. When all processing is complete, output to the printer, the punch, and the XYTRAN output data block is accomplished.

4.63.8 Subroutines

4.63.8.1 Subroutine Name: XYDUMP

1. Entry Point: XYDUMP
2. Purpose: To perform phase II as described above.
3. Calling Sequence: CALL XYDUMP (IARG,ITYPE).

   IARG - 201, GINØ output data block number.

   ITYPE - 1 for RESPØNSE, 2 for PSDF, 3 for AUTØ.

4.63.8.2 Subroutine Name: XYFIND

1. Entry Point: XYFIND
2. Purpose: To position one of the XYTRAN input data blocks (2 thru 6) to the beginning of a data set record for a particular ELEMENT-ID or POINT-ID of a specific

data type.

3. Calling Sequence: CALL XYFIND ($\$n_1$,$\$n_2$,$\$n_3$,MAJID,IDZ)

n_1 = Return taken in the event an end-of-file is sensed when an EØF should not be hit.

n_2 = Return taken in the event an end-of-record is sensed when an end-of-record. should not be hit.

n_3 = Return taken if the data requested could not be found.

MAJID = An array of the eleven data type major-IDs.

IDZ = Pointer into the Z array of open core to an ELEMENT-ID or POINT-ID.

4.63.8.3 Subroutine Name: XYØUT

1. Entry Point: XYØUT

2. Purpose: To output to the system printer unit an xy-output summary or to output to the system printer and/or punch unit(s) an xy-output coordinate pair.

3. Calling Sequence: CALL XYØUT (IARG,BUFF)

IARG = { <0 implies print summary. / ≥0 implies print and/or punch coordinate pair.

BUFF = Array containing data to be output.

4.63.8.4 Subroutine Name: XYLØG

1. Entry Point: XYLØG

2. Purpose: To analyze the input arguments V1 and V2 and to reset these arguments to powers of ten bracketing the original values. An example follows.

Input arguments.
- V1 = 0.5
- V2 = 5.6
- IARG = Undefined

Output arguments.
- V1 = 0.1
- V2 = 10.0
- IARG = 2

3. Calling Sequence: CALL XYLØG(V1,V2,IARG)

V1 = Smaller input real variable.

V2 = Larger input real variable.

IARG = Number of logrithmic cycles needed to bracket V1 and V2. (Set by XYLØG before return)

4.63.8.5 Subroutine Name: XYTICS

1. Entry Point: XYTICS

2. Purpose: To accept user-specified xy-plot edge-tic specifications and compute actual edge-tic beginning and ending values, their increments to the successive edge-tics, and their scientific values with powers of ten.

3. Calling Sequence: CALL XYTICS (IØUT,ØUT,IARG1,R1,R2,ISKIP)

IØUT = Integer output array
ØUT = Real output array
} One and the same array.

IARG1 = Number of edge-tic divisions desired by user.

R1 = Minimum coordinate value of edge.

R2 = Maximum coordinate value of edge.

ISKIP = Edge-tic skip count indicating which edge-tics are to have a value printed along with the tic-mark.

4.63.9 Design Requirements

1. The XYTRAN design requires that for a particular frame all of the curve data for the curves of that frame fit in core. If this condition is not possible, one curve at a time will be cancelled, with a warning message output, until the condition is met for the frame in question.

2. The following CØMMØN blocks are used in the subroutines of module XYTRAN.

a. CØMMØN/XYWØRK/

This common block contains variables required in the processing of the user output requests.

b. CØMMØN/XYTRZZ/

Defines open core for the module.

4.63.10 Diagnostic Messages

All XYTRAN diagnostic messages are of a USER-WARNING nature. There are no FATAL type error diagnostics. XYTRAN is in all cases expected to make a normal return.

4.64 FUNCTIONAL MODULE RANDØM (RANDOM ANALYSIS MODULE)

4.64.1 Entry Point: RANDØM

4.64.2 Purpose

To compute power spectral density functions and autocorrelation functions from frequency response data.

4.64.3 DMAP Calling Sequence

```
RANDØM    XYCDB,DIT,PSDL,ØUPVC2,ØPP2,ØQP2,ØESC2,ØEFC2,CASECC/PSDF,AUTØ/V,N,NØRAND  $
```

4.64.4 Input Data Blocks

XYCDB - XY Plotter Control Data Block.

DIT - Direct Input Tables.

PSDL - Power Spectral Density List.

ØUPVC2 - Output displacement vector requests (p set, SØRT2, complex).

ØPP2 - Output load vector requests (p set, SØRT2, complex).

ØQP2 - Output forces of single-point constraint (p set, SØRT2, complex).

ØESC2 - Output element stress requests (SØRT2, complex).

ØEFC2 - Output element force requests (SØRT2, complex).

CASECC - Case Control Data Table.

Notes:
1. If XYCDB is purged, RANDØM returns.
2. DIT cannot be purged if PSDL points to tables in DIT.
3. If PSDL is purged, RANDØM returns.
4. ØUPVC2, ØPP2, ØQP2, ØESC2, ØEFC2 must contain the requested outputs.
5. CASECC cannot be purged.

4.64.5 Output Data Blocks

PSDF - Power Spectral Density Table.

AUTØ - Autocorrelation function table.

Notes: PSDF and AUTØ cannot be purged.

4.64.6 Parameters

NØRAND - Output-integer-no default. NØRAND = -1, if no random analysis is requested; 0, otherwise.

4.64.7 Method

4.64.7.1 Overview of the Method

The Random Analysis Module calculates power spectral density functions, autocorrelation functions and mean deviations for selected displacements, loads, forces of single-point constraint, and element forces and stresses.

4.64.7.2 Module Initialization

The following 4 steps of subroutine RAND7 comprise module initialization.

1. The XYCDB must be present or RANDØM returns.
2. A set of RANDPS Bulk Data cards from PSDL must be selected in CASECC or RANDØM returns.
3. The frequency list is extracted from the first non-empty data file.
4. The selected RANDPS cards are read in and stored. The tables referenced are prepared by subroutine PRETAB. The RANDPS (see section 2.4 of the User's Manual) card defines the functions

$$S_{ab}(f) = (x + iy)\, F_K(f) \qquad (1)$$

where a is the subcase id of the excited load set; b is the subcase id of the applied load set ($a \leq b$); (x,y) is a complex number such that if a = b, then y must be 0.0; and K is the table identification number of a TABRND1 Bulk Data card which defines $F_k(f)$, a power spectral density as a tabular function of frequency.

If on any RANDPS card $a \neq b$, the equations are called coupled, otherwise they are called uncoupled.

4.64.7.3 The Uncoupled Case

The following eight steps are accomplished in subroutine RAND5.

1. The XYCDB is read for a list of requested points. This list is stored in core. (Subroutine RAND6).

2. Core is allocated for as many points as possible at one word per frequency. If all points will not fit in core, another pass will be made on this file.

3. Compute $S_{aa}(f)$ at each load change (subroutine TAB).

4. Read in the data from the SØRT2 data block and compute:

$$S_{ja}(f) = |U_j(f)|^2 S_{aa}(f), \tag{2}$$

where $U_j(f)$ is the response of the j^{th} point at frequency f.

5. These are summed over all loads to form the power spectral density function:

$$S_j(f) = \sum_a S_{ja}(f), \tag{3}$$

where 'a' runs over all subcase id's on the RANDPS cards.

6. When all subcases for the points in core have been processed, the mean response $\bar{q}_j$ is calculated in subroutine RAND3 for each point j:

$$\bar{q}_j = \left\{\frac{1}{2} \sum_{i=1}^{N-1} [(S_j(f_i) + S_j(f_{i+1})] (f_{i+1} - f_i)\right\}^{1/2}, \tag{4}$$

where N = number of frequencies. The mean response is output with both the PSDF and the autocorrelation function.

7. If PSDF for point j is requested, one id and data record are written on the PSDF data block.

8. If an autocorrelation function is requested for point j, the $S_j(f)$ are transformed to the time domain to give the autocorrelation function:

$$R_j(\tau_m) = \sum_{i=1}^{N-1} \left\{ \frac{1}{4\pi^2\tau^2_m} \left[\frac{S_j(f_{i+1}) - S_j(f_i)}{(f_{i+1} - f_i)}\right] [\cos(2\pi\tau_m f_{i+1}) - \cos(2\pi\tau_m f_i)] + \right.$$

$$\left. \frac{1}{2\pi\tau_m} [S_j(f+1) \sin(2\pi\tau_m f_{i+1}) - S_j(f_i) \sin(2\pi\tau_m f_i)] \right\}, \tag{5}$$

where i is the index of the frequencies; N is highest frequency; τ_m is defined by

$$\tau_m = \tau_o + \frac{m}{M}(\tau_{max} - \tau_o), \tag{6}$$

where τ_o is the starting time lag, M is the number of time lag intervals, and τ_{max} is the maximum time lag ($o < \tau_o < \tau_m$), all of which are defined on a RANDT1 Bulk Data card. Note that if, in Equation 5, $\tau_m = 0$, then

$$R_j(\tau_m) = \overline{q}_j^2. \tag{7}$$

If more points for this data block remain to be done, the file is rewound and another pass is made. If additional file types are requested, steps 1 through 8 outlined above are repeated. This completes the uncoupled case processing.

4.64.7.4 The Coupled Case

The following 6 steps are accomplished in subroutine RAND8.

1. A list of unique subcase id's is extracted from the RANDPS cards.

2. The XYCDB is read for a list of requested points. This list is stored in core (subroutine RAND6).

3. An array of core is reserved for each point as follows:
Let NFREQ = the number of frequencies used and NUNØ be the number of unique subcase id's mentioned on the RANDPS cards. Each point requires 2 NFREQ*NUNØ words of storage.

As many points as possible are done at once. The data file is read and the data are stored (real/imaginary) for each point until all subcases for all points in core have been processed.

4. For each RANDPS card $S_{ab}(f_i)$ is looked up for all f (subroutine TAB).
For each point in core $S_j^1(f)$ is computed:

$$S_j^1(f) = H_{ja}(f)\, S_{ab}(f)\, \overline{H}_{jb}(f), \tag{8}$$

where $H_{ja}(f)$ denotes the value of point j for subcase a. The bar over the third factor in Equation 8 denotes the complex conjugate. These $S_j^1(f)$ are summed over all RANDPS cards to form $S_j(f)$:

$$S_j(f) = \left|\sum_{ab}\sum H_{ja}(f)\, S_{ab}(f)\, \overline{H}_{jb}(f)\right|. \qquad (9)$$

Note that $S_{ba} = \overline{S}_{ab}$, the complex conjugate.

5. The mean response and autocorrelation functions are computed as in Equations 5, 6 and 7.

6. If more points for this file remain to be done, the file is rewound and another pass is made.

If additional file types are requested, steps 1 thru 6 are repeated. If not, the coupled case processing is complete.

4.64.8 Subroutines

4.64.8.1 Subroutine Name: RAND7

1. Entry Point: RAND7

2. Purpose: To initialize for both the coupled and uncoupled cases.

3. Calling Sequence: CALL RAND7(IFILE,NFILE,PSDL,DIT,ICØUP,NFREQ,NPSDL,NTAU,LTAB, CASECC,XYCDB).

PSDL,DIT,CASECC,XYCDB are GINØ file numbers for their respective data blocks - - integer - input.

IFILE - Array of GINØ file numbers of data files to RANDØM - integer - input.

NFILE - Number of files in IFILE - integer - input.

ICØUP - -1 No RANDØM analysis to be done.

- 0 uncoupled algorithm to be used - integer - output.

- 1 coupled algorithm to be used.

NFREQ - Number of frequencies - integer - output.

NPSDL - Number of RANDPS cards selected - integer - output.

NTAU - Number of τ's on RANDT1 cards - integer - output.

LTAB - Amount of core taken up by table storage - integer - output.

CØMMØN/RANDMX/

RAND7 stores most of its output data in /RANDMX/. See core storage layout of /RANDMX/ (section 4.64.9).

4.64.8.2 Subroutine Name: RAND5

1. Entry Point: RAND5

2. Purpose: To compute uncoupled PSDF and AUTØ numbers.

3. Calling Sequence: CALL RAND5(NFREQ,NPSDL,NTAU,XYCDB,LTAB,IFILE,PSDF,AUTØ,NFILE)

PSDF,AUTØ - GINØ file numbers of respective files - integer - input.

Other variables are as in RAND7 (section 4.64.8.1).

4.64.8.3 Subroutine Name: RAND8

1. Entry Point: RAND8

2. Purpose: To compute coupled PSDF and AUTØ numbers.

3. Calling Sequence: CALL RAND8 (Same as RAND5).

4.64.8.4 Subroutine Name: RAND1

1. Entry Point: RAND1

2. Purpose: To put one ØFP type ID on PSDF and AUTØ.

3. Calling Sequence: CALL RAND1 (FILE,MID,TYPE,ID,CØMP,Q).

FILE - GINØ file number of output file - integer - input.

MID - File type (PSDF = 4001,AUTØ = 4002) - integer - input.

TYPE - Curve type - DISP,VELØ,ACCE,LØAD,SPLF,ELFØ, or STRE - BCD, input.

ID - Point id - integer - input.

CØMP - Point component - integer - input.

Q - Mean deviation - real - input.

4.64.8.5 Subroutine Name: RAND2

1. Entry Point: RAND2

2. Purpose: To read a SØRT2 type output file until it finds a point id selected by the user in a list.

3. Calling Sequence: CALL RAND2 (FILE,ILIST,LØAD,IF,LEN,LLIST)

FILE - GINØ file number of the SØRT2 data file - integer - input.

ILIST - List of user desired points - input and output.

LØAD - Subcase id of first data record in ILIST - integer - output.

IF - Format of data - real/imaginary or magnitude/phase - integer - output.

LEN - Length of the data line for this record - integer - output.

LLIST - Length of the ILIST array.

4.64.8.6 Subroutine Name: RAND3

1. Entry Point: RAND3

2. Purpose: To compute the mean response $\overline{q}$.

3. Calling Sequence: CALL RAND3 (F,S,Q,N)

F - Array of frequencies - real - input.

S - Array of power spectral density functions - real - input.

Q - Mean response - real - output.

N - Length of the F and S arrays - integer - input.

4.64.8.7 Subroutine Name: RAND4

1. Entry Point: RAND4

2. Purpose: To compute the autocorrelation function $R(\tau)$.

3. Calling Sequence: CALL RAND4 (F,S,TAU,R,N)

F,S,N are as described in RAND3.

TAU - τ point at which R is to computed - real - input.

R - Autocorrelation function at TAU - real - output.

4.64.8.8 Subroutine Name: RAND6

1. Entry Point: RAND6

2. Purpose: To extract from the XYCDB a list of user requested points for RANDØM output.

3. Calling Sequence: CALL RAND6 (XYCDB,BUFFER,NPØINT,IZ,INPUT)

XYCDB - GINØ file number of the XYCDB data block - integer - input.

BUFFER - GINØ buffer - array - input.

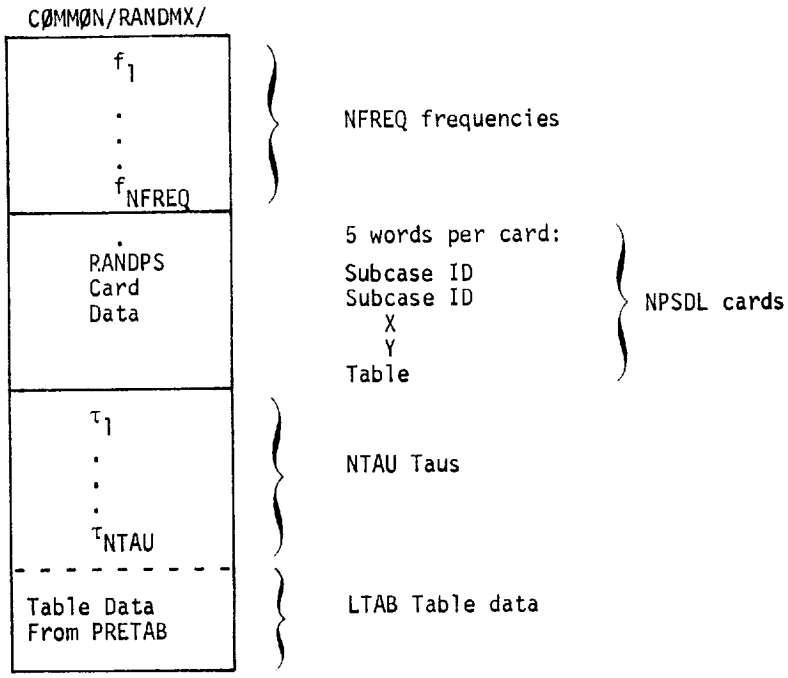
NPØINT - Number of points requested by the user for this file - integer - output.

IZ - Array in which RAND6 stores the list of requests - integer - output.

INPUT - GINØ file number of data file for which list of request is desired.

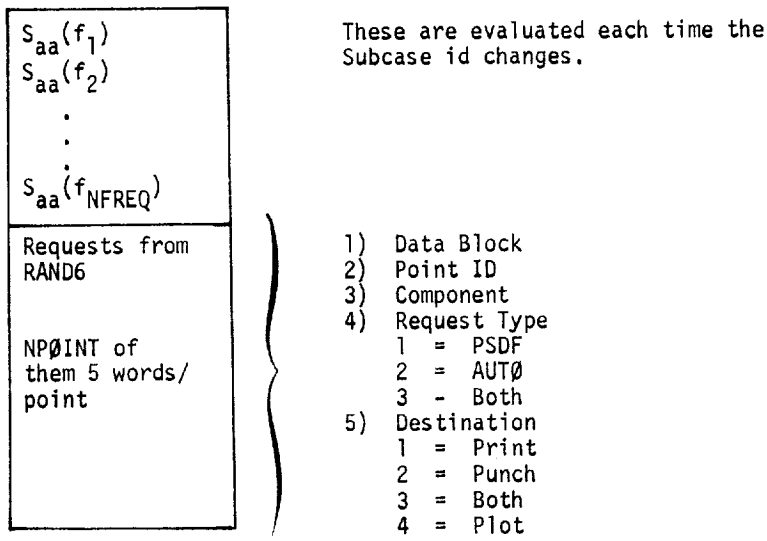
4.64.9 Design Requirements

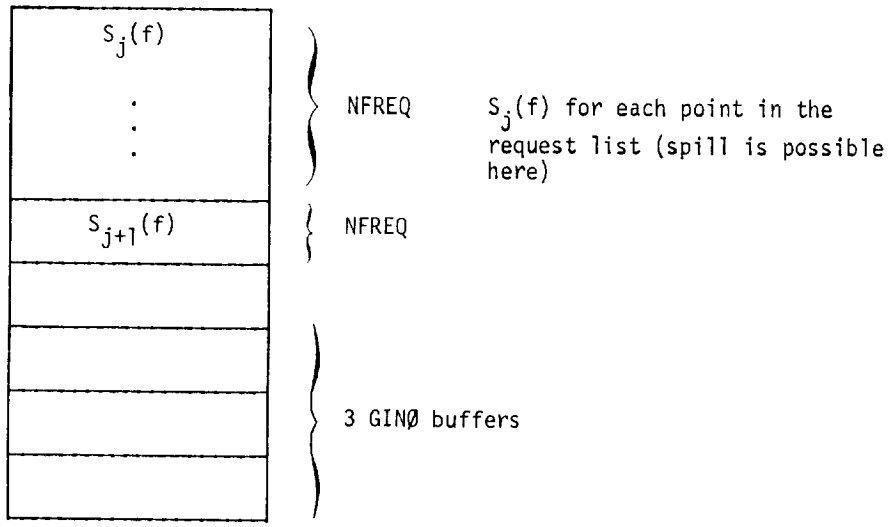
Open Core at /RANDMX/ is arranged as follows:



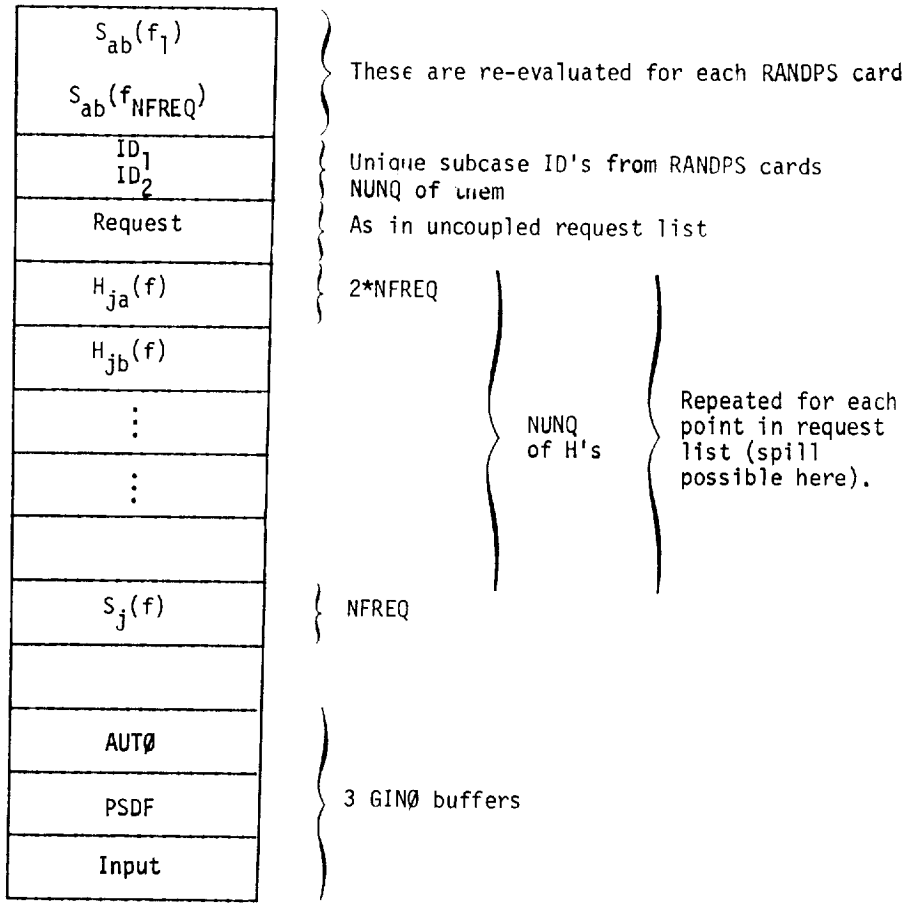
The above data are placed in core by RAND7 and are the same for both the coupled and uncoupled cases. The remaining data are core dependent.

Uncoupled case data:





Coupled case data:



4.64.10 Diagnostic Messages

RANDØM is defined as an output processor and thus must not stop due to user input error. Hence all messages are of a warning nature.

Random may issue message 3048.

4.65 FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

4.65.1 Entry Point: TRD

4.65.2 Purpose

To solve the transient problem.

4.65.3 DMAP Calling Sequence

```
TRD     CASECC,USETD,DLT,TRL,NLFT,DIT,KHH,BHH,MHH,GMD,GØD,PHIDH/UHVT,PDT,PST,PPT,PNLH/V,N,
        FØRM/V,N,LUSETD/V,N,MPCF1/V,N,SINGLE/V,N,ØMIT/V,N,NØNCUP/V,N,NØUE     $
```

4.65.4 Input Data Blocks

CASECC - Case Control Data table.

USETD - Displacement set definitions table-dynamics.

DLT - Dynamic Loads Table.

TRL - Transient Response List.

NLFT - Non-Linear Forcing Table.

DIT - Direct Input Tables.

KHH - Modal stiffness matrix - h set.

BHH - Modal damping matrix - h set.

MHH - Modal mass matrix - h set.

GMD - Multipoint constraint transformation matrix - dynamics.

GØD - Omitted coordinate transformation matrix - dynamics.

PHIDH - Transformation Matrix from d - set to modal coordinates.

Notes:

1. CASECC cannot be purged.
2. USETD cannot be purged.
3. DLT cannot be purged if a dynamic load was selected in CASECC.
4. TRL cannot be purged.

5. NLFT cannot be purged if nonlinear loads are selected in CASECC.

6. GMD cannot be purged if MPCF1 $\geq$ 0.

7. GØD cannot be purged if ØMIT $\geq$ 0.

8. PHIDH cannot be purged if FØRM = MØDAL.

4.65.5 Output Data Blocks

UHVT - Modal transient solution vectors - h set.

PDT - Linear dynamic load matrix for transient analysis - d set.

PST - Linear load vector for transient analysis - s set.

PPT - Linear dynamics loads for transient analysis - p set.

PNLH - Nonlinear loads in modal transient problem - h set.

Notes:

1. UHVT cannot be purged.

2. PDT cannot be purged.

3. PST cannot be purged if SINGLE $\geq$ 0.

4. PPT cannot be purged.

5. PNLH cannot be purged if nonlinear loads are selected.

4.65.6 Parameters

FØRM - Input-BCD-no default. If FØRM = MØDAL a modal formulation will be used, otherwise a direct formulation will occur.

LUSETD - Input-integer-no default. LUSETD indicates length of p set.

MPCF1 - Input-integer-no default. If MPCF1 $\geq$ 0 multipoint constraints are present.

SINGLE - Input-integer-no default. If SINGLE $\geq$ 0 single-point constraints are present.

ØMIT - Input-integer-no default. If ØMIT $\geq$ 0 omitted coordinates are present.

NØNCUP - Input-integer-no default. If NØNCUP = -1 an uncoupled solution will be done.

NØUE - Input-integer-no default. NØUE indicates the number of extra points-used in nonlinear load formulation.

4.65.7 Method

4.65.7.1 Overview of the Method

The Transient Analysis module integrates, over specified time periods, equations of motion of a structure having time dependent loads. A general structure may be used with real stiffness, mass and damping matrices. Non-linear effects may be calculated by specifying certain loading functions on the free, physical displacements of the system. This analysis is particularly useful when shock loads are applied to a structure. It is also more efficient than frequency analysis or complex eigenvalue analysis when the applied loads are well defined and the frequency characteristics are secondary to damping and peak load characteristics. This analysis is also the only dynamic general system analysis which allows non-linearities.

4.65.7.2 Logical Phases of Solution

1. The applied load vectors are assembled. The time increment from the TRL data block is used to identify the times at which the loads are assembled. The initial conditions are assembled.

2. The loads are reduced to the solution coordinates.

3. The left hand matrix of the general integration equation, Equation 15 below, and the two right hand matrices are assembled. The triangular decomposition of the left hand matrix is performed.

4. The solution loop of the program may now proceed until the time increment is changed. The steps are:

    a. Compute the non-linear load for this time step. Add this load to the load vectors computed in Step 2.

    b. Multiply the displacement vectors into the right hand matrices and add the resultant vectors to the applied load vector.

    c. Solve for the left hand displacement vector by performing a back substitution into the triangular decomposition of the left hand matrix. If this is an output time step, the velocity and acceleration are computed using differences of the displacement vectors.

d. If the time increment changes for the next time step, the program returns to step 3. If the increment is the same, steps 4a thru 4d are repeated.

5. If the equations are in the uncoupled modal formulation form (i.e., no transfer functions, direct input matrices, or non-linear functions), the solution logic is much faster. For each coordinate, the displacement, velocity and acceleration may be computed independently versus time. Steps 3 and 4 are omitted.

4.65.7.3 Algorithms for Each Logical Phase

1. Assembly of Applied load vectors: CASECC is read and the selected time step, load, non-linear load, and initial condition set ids are extracted. The initial displacement and velocity vectors are packed on a scratch file for later use. The computation and output data on the TSTEP card are stored in core. The DLT is read to build a list of load "id's", and table "id's" and scale factors for the simple components of a load. Core is zeroed, and loads for as many times as possible are constructed in core and output. Two types of loads are available, TLØAD1 and TLØAD2.

For the TLØAD1 load,

$$P(t) = AF(t-\tau), \tag{1}$$

where F is a user input table.

For the TLØAD2 load,

$$P(t) = \begin{cases} At^{\beta}e^{c\tilde{t}} \cos(2\pi f \tilde{t} + P) & 0 \le \tilde{t} \le T2-T1 \\ 0 & \text{elsewhere} \end{cases} \tag{2}$$

where $t = \tilde{t} - T_1 - \tau$ and T_1, T_2, f, P, C and β are user input coefficients.

(Subroutines TRD1A, PRETAB and TAB).

2. Reduction of loads to the solution coordinates:

If m's are present (MPCF1$\geq$0),

$$\{P_p\} \Rightarrow \begin{Bmatrix} \bar{P}_{ne} \\ P_m \end{Bmatrix}, \tag{3}$$

$$\{P_{ne}\} = [G_o^d]^T \{P_m\} + \{\bar{P}_{ne}\}. \tag{4}$$

If s's are present (SINGLE $\geq$ 0),

$$\{P_{ne}\} \Longrightarrow \left\{ \frac{P_{fe}}{P_s} \right\}. \tag{5}$$

$\{P_s\}$ is reduced to the output times and output on data block PS.

If o's are present (ØMIT $\geq$ 0),

$$\{P_{fe}\} \Longrightarrow \left\{ \frac{\bar{P}_d}{P_o} \right\}, \tag{6}$$

$$\{P_d\} = [G_o^d]^T \{P_o\} + \{\bar{P}_d\}. \tag{7}$$

$\{P_d\}$ is reduced to the output times and output on data block PØ.

If this is a modal formulation (FØRM = MØDAL),

$$\{P_h\} = \{\Phi_{dn}\}^T \{P_d\}. \tag{8}$$

$\{P_p\}$ is reduced to output times and output on data block PP.

3. Solution of the coupled equations: The matrix

$$[D] = \left(\frac{1}{\Delta t^2} [M] + \frac{1}{2\Delta t} [B] + \frac{1}{3} [K] \right), \tag{9}$$

is formed and decomposed. The matrix

$$[C] = \left(\frac{2}{\Delta t^2} [M] - \frac{1}{3} [K] \right), \tag{10}$$

is formed and saved. The matrix

$$[E] = \left[\frac{-1}{\Delta t^2} [M] + \frac{1}{2\Delta t} [B] - \frac{1}{3} [K] \right], \tag{11}$$

is formed and saved.

The solution loops then proceed until a time step change occurs. The initial conditions are brought in and the starting equation

$$[D]\{u_1\} = \frac{1}{3}\{P^1_{-1} + P^1_0 + P_1\} + \{N_0\} + [C]\{u_0\} + [E]\{u^1_{-1}\}, \tag{12}$$

is solved for $\{u_1\}$, where $\{u_0\}$ is the starting displacement vector;

$$\{u^1_{-1}\} = \{u_0\} - \{\dot{u}_0\}\,\Delta t, \tag{13}$$

$\{\dot{u}_0\}$ is the starting velocity vector;

$$\{P^1_0\} = [K]\{u_0\} + [B]\{\dot{u}_0\}; \tag{14}$$

$\{N_0\}$ is the non-linear load calculated from $\{u_0\}$; and

$$\{P^1_{-1}\} = [K]\{u^1_{-1}\} + [B]\{\dot{u}_0\} = \{P^1_0\} - \Delta t\,[K]\{\dot{u}_0\}. \tag{15}$$

Note that the load computed at t = 0 is never used but is replaced by $\{P^1_0\}$. $\{u_2\}$ through $\{u_n\}$ are now computed from the general equation:

$$[D]\{u_{i+2}\} = \frac{1}{3}\{P_i + P_{i+1} + P_{i+2}\} + \{N_{i+1}\} + [C]\{u_{i+1}\} + [E]\{u_i\}. \tag{16}$$

If non-linear loads are selected, they are evaluated directly at the solution points for time step by the following process. NØLIN1 loads are computed as,

$$P_i(t) = S\,T\,(u_j(t)), \tag{17}$$

where T is a user selected table, i is the loaded solution point, j is the deflecting point, u_j is the previously computed displacement at point j. NØLIN2 loads are computed as,

$$P_i(t) = S\,u_j\,(t)\,u_k\,(t), \tag{18}$$

where i, j, and k are as in NØLIN1 loads.

NØLIN3 loads are computed as,

$$P_i(t) = \begin{cases} S\ \{u_j(t)\}A, & u_j(t)>0 \\ 0, & u_j(t)\leq 0 \end{cases} \tag{19}$$

NØLIN4 loads are computed as,

$$P_i(t) = \begin{cases} -S\ \{-u_j(t)\}A, & u_j(t)< 0 \\ 0, & u_j(t) \geq 0. \end{cases} \tag{20}$$

The user specifies the set of times at which data is to be saved. If the current time is an output time, the displacement vector for time $t = t_i$ is output.

The velocity vector given by:

$$\{\dot{u}_i\} = \frac{1}{2\Delta t}\ [\{u_{i+1}\} - \{u_{i-1}\}], \tag{21}$$

is output.

The acceleration vector given by

$$\{\ddot{u}_i\} = \frac{1}{\Delta t_i^2}\ [\{u_{i+1}\} + \{u_{i-1}\} - 2\ \{u_i\}], \tag{22}$$

is output.

If the time step is scheduled to change at t_{i+1} from Δt_1 to Δt_2, the displacement for time i+1 has been calculated. $\{u_{i-1}\}$, $\{u_i\}$, and $\{u_{i+1}\}$ are saved along with $\{P_{i+1}\}$. The matrices are formed and decomposed as in Equations 9,10, and 11 for $\Delta t = \Delta t_2$.

The following equation is used for computing $\{u_{i+2}\}$,

$$[D]\ \{u_{i+2}\} = \frac{1}{3}\ \{P_i^1 + P_{i+1} + P_{i+2}\} + \{N_{i+1}\} + [C]\ \{u_{i+1}\} + [E]\ \{u_i^1\}. \tag{23}$$

The vectors $\{P_i^1\}$ and $\{u_i^1\}$ in the above equation are calculated as follows. Define

$$\{\dot{u}_{i+1}\} = \frac{1}{\Delta t_1} (\{u_{i+1}\} - \{u_i\}), \tag{24}$$

$$\{\ddot{u}_{i+1}\} = \frac{1}{\Delta t_1^2} (\{u_{i+1}\} - 2\{u_i\} + \{u_{i-1}\}), \tag{25}$$

$$\{u_i^1\} = \{\dot{u}_{i+1}\} - \{\ddot{u}_{i+1}\} \Delta t_2, \tag{26}$$

then:

$$\{u_i^1\} = \{u_{i+1}\} - \Delta t_2 \{\dot{u}_{i+1}\} + \frac{\Delta t_2^2}{2} \{\ddot{u}_{i+1}\}, \tag{27}$$

$$\{P_i^1\} = [M] \{\ddot{u}_{i+1}\} + [B] \{\dot{u}_i^1\} + [K] \{u_i^1\}. \tag{28}$$

4. Solution of Uncoupled Modal Equation: If the method of matrix formulation is modal and no transfer functions or direct input matrices are used, the equations may be solved in a more accurate, more direct manner. The diagonal terms of MHH, BHH, and KHH are stored in core. The following data are necessary to solve the transient behavior of a modal coordinate (ξ_i).

m_i = Modal mass of mode (MHH)

b_i = Modal damping coefficient (BHH)

K_i = Modal stiffness (KHH)

$$\omega_{oi} = (K_i/m_i)^{1/2}, \tag{29}$$

$$\beta_i = \frac{b_i}{2m_i}, \tag{30}$$

$$\omega_i^2 = |\omega_{oi}^2 - \beta^2|, \tag{31}$$

t_j = time of the j^{th} time step,

h_j = time increment after the j^{th} time,

f_{ij} = applied load on coordinate i at the j^{th} time.

The following coefficients are generated for each distinct time increment and stored in core.

There are four cases. ($\varepsilon = 10^{-5}$ and the subscript i is implied).

a. If $\omega_o^2 > \beta^2 + \varepsilon$ (underdamped):

$$F = e^{-\beta h} (\cos \omega h + \frac{\beta}{\omega} \sin \omega h) , \tag{32}$$

$$G = \frac{1}{\omega} e^{-\beta h} \sin \omega h , \tag{33}$$

$$A = \frac{1}{hk\omega} \{e^{-\beta h} [(\frac{\omega^2-\beta^2}{\omega_o^2} - \beta h) \sin \omega h - (\frac{2\omega\beta}{\omega_o^2} + h\omega) \cos \omega h] + \frac{2\beta\omega}{\omega_o^2}\} , \tag{34}$$

$$B = \frac{1}{hk\omega} \{e^{-\beta h} [(- \frac{\omega^2-\beta^2}{\omega_o^2}) \sin \omega h + \frac{2\omega\beta}{\omega_o^2} \cos \omega h] + \omega h - \frac{2\beta\omega}{\omega_o^2}\} , \tag{35}$$

$$F' = - \frac{\omega_o^2}{\omega} e^{-\beta h} \sin \omega h , \tag{36}$$

$$G' = e^{-\beta h}(\cos \omega h - \frac{\beta}{\omega} \sin \omega h) , \tag{37}$$

$$A' = \frac{1}{hk\omega} [e^{-\beta h} \{(\beta + h\omega_o^2) \sin \omega h + \omega \cos \omega h\} - \omega] , \tag{38}$$

$$B' = \frac{1}{hk\omega} [-e^{-\beta h} (\beta \sin \omega h + \omega \cos \omega h) + \omega] , \tag{39}$$

b. If $|\omega_o^2 - \beta^2| < \varepsilon$ (critically damped):

$$F = e^{-\beta h}(1 + \beta h) , \tag{40}$$

$$G = he^{-\beta h} , \tag{41}$$

$$A = \frac{1}{hk}\left[\frac{2}{\beta} - \frac{1}{\beta} e^{-\beta h} (2 + 2h\beta + h^2 \beta^2)\right] , \tag{42}$$

$$B = \frac{1}{hk\beta}\left[-2 + \beta h + e^{-\beta h} (2 + \beta h)\right] , \tag{43}$$

$$F' = -\beta^2 h e^{-\beta h} , \tag{44}$$

$$G' = e^{-\beta h}(1 - \beta h) , \tag{45}$$

$$A' = \frac{1}{hk}\left[e^{-\beta h} (1 + h\beta + h^2\beta^2) - 1\right] , \tag{46}$$

$$B' = \frac{1}{hk}\left[1 - e^{-\beta h} (\beta h + 1)\right] . \tag{47}$$

c. If $\omega_o^2 < \beta^2 - \epsilon$ (over damped):

$$F = e^{-\beta h} (\cosh \omega h + \frac{\beta}{\omega} \sinh \omega h) , \tag{48}$$

$$G = \frac{1}{\omega} e^{-\omega h} \sinh \omega h , \tag{49}$$

$$A = \frac{1}{hk\omega}\left\{e^{-\beta h}\left[\left(\frac{\omega^2 + \beta^2}{\omega_o^2} - h\beta\right) \sinh \omega h - \left(\frac{2\omega\beta}{\omega_o^2} + h\omega\right) \cosh \omega h\right] + \frac{2\omega\beta}{\omega_o^2}\right\} , \tag{50}$$

$$B = \frac{1}{hk\omega}\left\{e^{-\beta h}\left[\frac{\omega^2 + \beta^2}{\omega_o^2} \sinh \omega h + \frac{2\omega\beta}{\omega_o^2} \cosh \omega h\right] + \omega h - \frac{2\beta\omega}{\omega_o^2}\right\} , \tag{51}$$

$$F' = -\frac{\omega_o^2}{\omega} e^{-\beta h} \sinh \omega h , \tag{52}$$

$$G' = e^{-\beta h} (\cosh \omega h - \frac{\beta}{\omega} \sinh \omega h) , \tag{53}$$

$$A' = \frac{1}{hk\omega}\left[e^{-\beta h} \left\{(\beta + h\omega_o^2) \sinh \omega h + \omega \cosh \omega h\right\} - \omega\right] , \tag{54}$$

$$B' = \frac{1}{hk\omega}\left[- e^{-\beta h} (\beta \sinh \omega h + \omega \cosh \omega h) + \omega\right] . \tag{55}$$

d. If $\omega_o = \beta = 0$ (undamped):

$$F = 1, \tag{56}$$

$$G = h, \tag{57}$$

$$A = h^2/3m, \tag{58}$$

$$B = h^2/6m, \tag{59}$$

$$F' = 0, \tag{60}$$

$$G' = 1, \tag{61}$$

$$A' = h/2m, \tag{62}$$

$$B' = h/2m. \tag{63}$$

The equations for each displacement, velocity, and acceleration in terms of the applied loads and previous displacement and velocity are:

$$\xi_{i,j+1} = F_i\xi_{i,j} + G_i\dot{\xi}_{i,j} + A_i f_{i,j} + B_i f_{i,j+1}\ , \tag{64}$$

$$\dot{\xi}_{i,j+1} = F'_i\xi_{i,j} + G'_i\dot{\xi}_{i,j} + A'_i f_{i,j} + B'_i f_{i,j+1}\ , \tag{65}$$

$$\ddot{\xi}_{i,j+1} = \frac{P_{i,j+1}}{m_i} + \frac{b_i\dot{\xi}_{i,j+1}}{m_i} - \frac{K_i\xi_{i,j+1}}{m_i}\ . \tag{66}$$

4.65.8 Subroutines

Utility routines PRETAB, TAB, SSG2A, CALCV, SSG2B, ADD, and DECØMP are used. See subroutine descriptions, section 3 for details.

4.65.8.1 Subroutine Name: TRD1A

1. Entry Point: TRD1A

2. Purpose: To assemble the loads at all time steps.

3. Calling Sequence: CALL TRD1A (DLT,TRL,CASECC,DIT,PPA,IC,LUSETD,NLFTP,NGRØUP, ITRL,MØDAL)

DLT, TRL, CASECC, DIT are GINØ file numbers of their respective data blocks - integer - input.

PPA - GINØ file number of applied loads - p set - for all times - integer - input.

IC - GINØ file number of initial condition matrix - integer - input.

LUSETD - Length of p set - integer - input.

NLFTP - Non-linear load set id selected in CASECC - integer - output.

NGRØUP - Number of time step changes - integer - output.

ITRL - Number of records to skip in TRL to obtain selected TSTEP card - integer - output.

MØDAL - If MØDAL = 1, a modal formulation is being used - integer - input.

4.65.8.2 Subroutine Name: TRD1B

1. Entry Point: TRD1B

2. Purpose: To reduce the applied loads to the solution set and remove all but output time loads.

3. Calling Sequence: CALL TRD1B (PPA,USETD,GMØ,GØD,PHIDH,MPCF1,SINGLE,ØMIT,FØRM, PD,PS,PP,PAPPLD,SCR1,SCR2,SCR3,SCR4,NGRØUP,TRL,ITRL)

PPA - Same as for TRD1A

USETD,GMØ, GØD,PHIDH,PD, PS,PP,TRL - GINØ file numbers of their respective data blocks - integer - input.

SCR1,..., SCR4 - GINØ file numbers of 4 scratch files.

MPCF1,SINGLE, ØMIT,FØRM - Module input parameters as explained in section 4.65.6.

NGRØUP,ITRL - Same as for TRD1A - integer - input.

PAPPLD - GINØ file number of applied loads - integer - input.

4.65.8.3 Subroutine Name: INITL

1. Entry Point: INITL

2. Purpose: To form [C] and [E] matrices and to form and decompose the [D] matrix.

3. Calling Sequence: CALL INITL (ØFFSET,DELTA)

CØMMØN/TRDXX/ See section 4.65.8.4.

ØFFSET - Length of reserved area of core - integer - input.

DELTA - Current time increment - real - input.

4.65.8.4 Subroutine Name: TRD1C

1. Entry Point: TRD1C

2. Purpose: To solve the coupled equations.

3. Calling Sequence: CALL TRD1C (IC,PAPPLD,NGRØUP,NLFTP,UDV,I,SCR1,DIT,NLFT,NØUE MØDA1,PNL)

IC,NGRØUP NLFTP - Are as described in TRD1A - integer - input.

UDV,DIT NLFT,PNL - Are GINØ file numbers of their respective data blocks - integer - input.

SCR1 - GINØ file number of a scratch file.

PAPPLD - Is as described in TRD1B.

NØUE - Module parameter.

MØDA1 - -1 if FØRM ≠ MØDAL. 1 if FØRM = MØDAL - integer - input.

I - Current loop count. Runs from 1 to number of time step changes - integer - input.

CØMMØN/TRDXX/IK(7)IM(7),IB(7),C,ULL,LLL,E,SCR1,SCR2,IØPEN

IK(7) - Matrix control block for K matrix.

IM(7) - Matrix control block for M matrix.

IB(7) - Matrix control block for B matrix.

C - GINØ file number for C matrix.

ULL,LLL - GINØ file numbers for decomposition products of D matrix.

E - GINØ file number for E matrix.

SCR1,SCR2- GINØ file numbers for 2 scratch files.

IØPEN - 1 implies C,ULL,LLL, and E are open.
0 implies C,ULL,LLL, and E are closed.

4.65.8.5 Subroutine Name: FØRM1

1. Entry FØRM1

2. Purpose: To compute $\{u_{-1}\}$, $\{P'_0\}$, and $\{P_{-1}\}$ for starting the integration procedure.

3. Calling Sequence: CALL FØRM1 (UØ,UDØTØ,UI,PØ,PI,DELTAT,IBUF)

UØ - Array of core containing $\{u_0\}$ - real - input.

UDØTØ - Array of core containing $\{\dot{u}_0\}$ - real - input.

UI - Array of core for storage of $\{u_{-1}\}$ - real - output.

PØ - Array of core for storage of $\{P'_0\}$ - real - output.

PI - Array of core for storage of $\{P_{-1}\}$ - real - output.

DELTAT - Current time step size - real - input.

IBUF - GINØ buffer

CØMMØN/TRDXX/ (see above).

4.65.8.6 Subroutine Name: MATVEC

1. Entry Point: MATVEC

2. Purpose: To form the product{X} = {X} + [A] {Y} where [A] is a matrix and {Y} is a vector.

3. Calling Sequence: CALL MATVEC (Y,X,FILEA,IBUF)

Y - Array of core containing Y array real - input.

X - Array of core containing X array real - input/output.

FILEA - Matrix control block for A. If FILEA(1) $\leq$ 0, MATVEC will return.

IBUF - GINØ buffer. If IBUF $\leq$ 0, MATVEC will assume the file is already in core.

COMMON/TPDXX/ (See section 4.65.8.4)

4.65.8.7 Subroutine Name: STEP

1. Entry Point: STEP

2. Purpose: To integrate forward 1 time step.

3. Calling Sequence: CALL STEP (U2,U1,UØ,P,IBUF)

U2 - Array which will contain $\{u_{i+2}\}$ - real - output.

U1 - Array containing $\{u_{i+1}\}$ - real - input.

UØ - Array containing $\{u_i\}$ - real - input.

P - Array containing combined load - real - input.

IBUF - GINØ buffer - input.

CØMMØN/TRDXX/ (See section 4.65.8.4)

4.65.8.8 Subroutine Name: INTFBS

1. Entry Point: INTFBS

2. Purpose: To perform the forward-backward substitution necessary to solve the system of equations: [A] {Y} = {X} for {Y}.

3. Calling Sequence: CALL INTFBS (X,Y,IBUF)

X - Load vector (i.e. right hand side) - real- input.

Y - Solution vector - real - output.

IBUF - GINØ buffer.

CØMMØN/TRDXX/ (See section 4.65.8.4)

CØMMØN/INFBS/FILEL(7),FILEU(7)

FILEL - Matrix control block of the lower triangular factor from the decomposition of A.

FILEU - Matrix control block of the upper triangular factor from the decomposition of A.

4.65.8.9 Subroutine Name: TRD1D

1. Entry Point: TRD1D

2. Purpose: To compute the non-linear loads at each time step.

3. Calling Sequence: CALL TRD1D

CØMMØN/TRDD1/NLFT,DIT,NLFTP,NØUT,ICØUNT,ILØØP,MØDA1,NZ,ICØRE,IU2,IP4,IPNL(7),NMØDES, NSTEP,PNL

The variables DIT,NLFT,NLFTP,MØDA1 and PNL are defined as in TRD1C (see section 4.65.8.4).

NØUT - Output interval - integer - input.

ICØUNT - Current time step counter - integer - input.

ILØØP - Current time change counter - integer - input.

NZ - Length of open core - integer - input.

ICØRE - Pointer to first unused cell of open core - integer - input.

IU2 - Pointer to displacement vector - 1 - integer - input.

IP4 - Pointer to load area -1 - integer - input.

IPNL - Matrix control block for PNL - integer - input/output.

NMØDES - Number of modes if modal formulation is being used - integer - input.

NSTEP - Number of times steps for this time increment - integer - input.

4.65.8.10 Subroutine Name: TRD1E

1. Entry Point: TRD1E

2. Purpose: To solve the uncoupled modal equations.

3. Calling Sequence: CALL TRD1E (MHH,BHH,KHH,PH,UHV,NGRØUP)

MHH,BHH,KHH, PH,UHV - GINØ file numbers of their respective data blocks - integer - input.

NGRØUP - Number of time step changes - integer - input.

4.65.8.11 Subroutine Name: FØRM2

1. Entry Point: FØRM2

2. Purpose: To compute $\{u_i'\}$ and $\{P_i'\}$ when changing time steps. (See Equations 23 through 28)

3. Calling Sequence: CALL FØRM2 (UDDIP1,UDIP1,UIP,PIP,IBUF)

UDDIP1 - Array of core containing $\{\ddot{u}_{i+1}\}$ - real - input.

UDIP1 - Array of core containing $\{\dot{u}_{i+1}\}$ - real - input.

UIP - Array of core containing $\{u_i'\}$ - real - output.

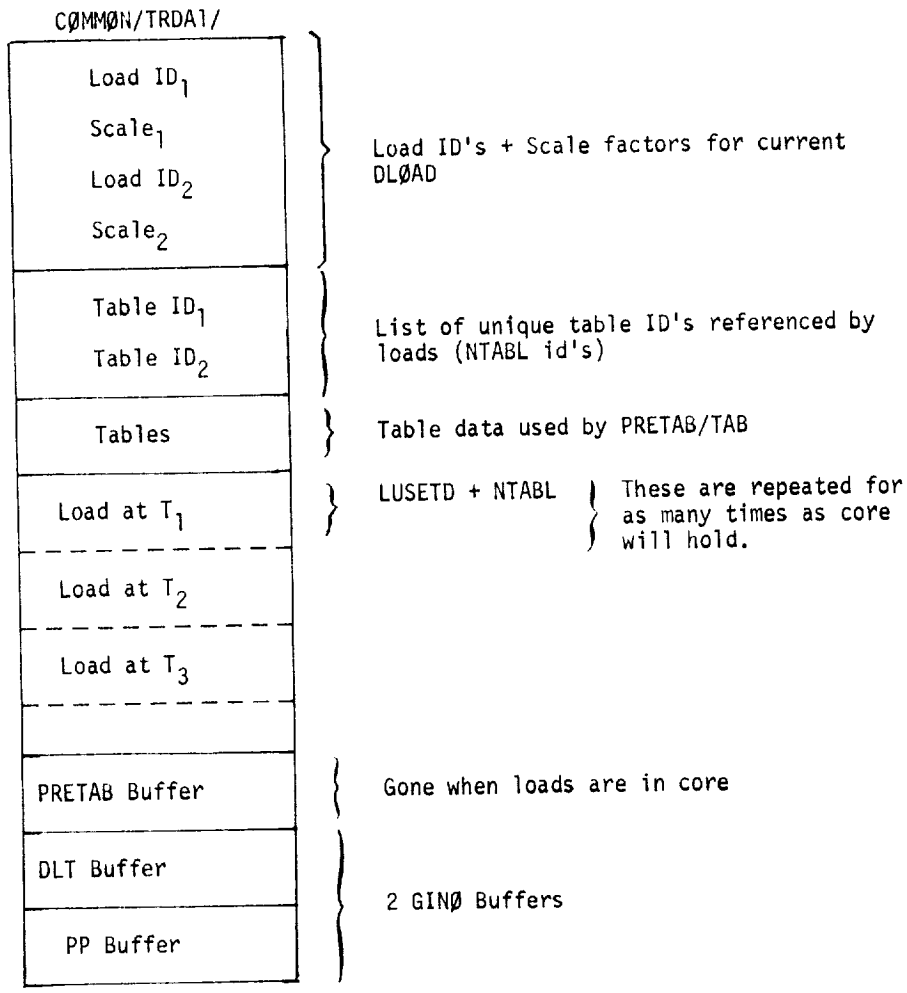
PIP - Array of core containing $\{P_i'\}$ - real - output.

IBUF - GINØ buffer

CØMMØN/TRDXX/ (See section 4.65.8.4)

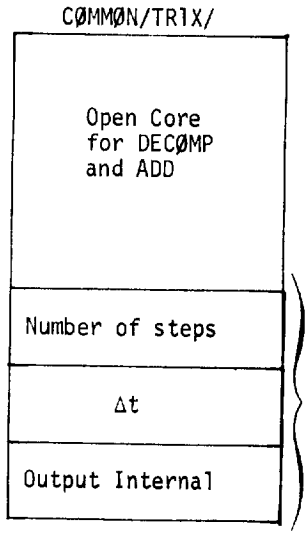
4.65.9 Design Requirements

1. Open core at /TRDA1/ is illustrated as follows:



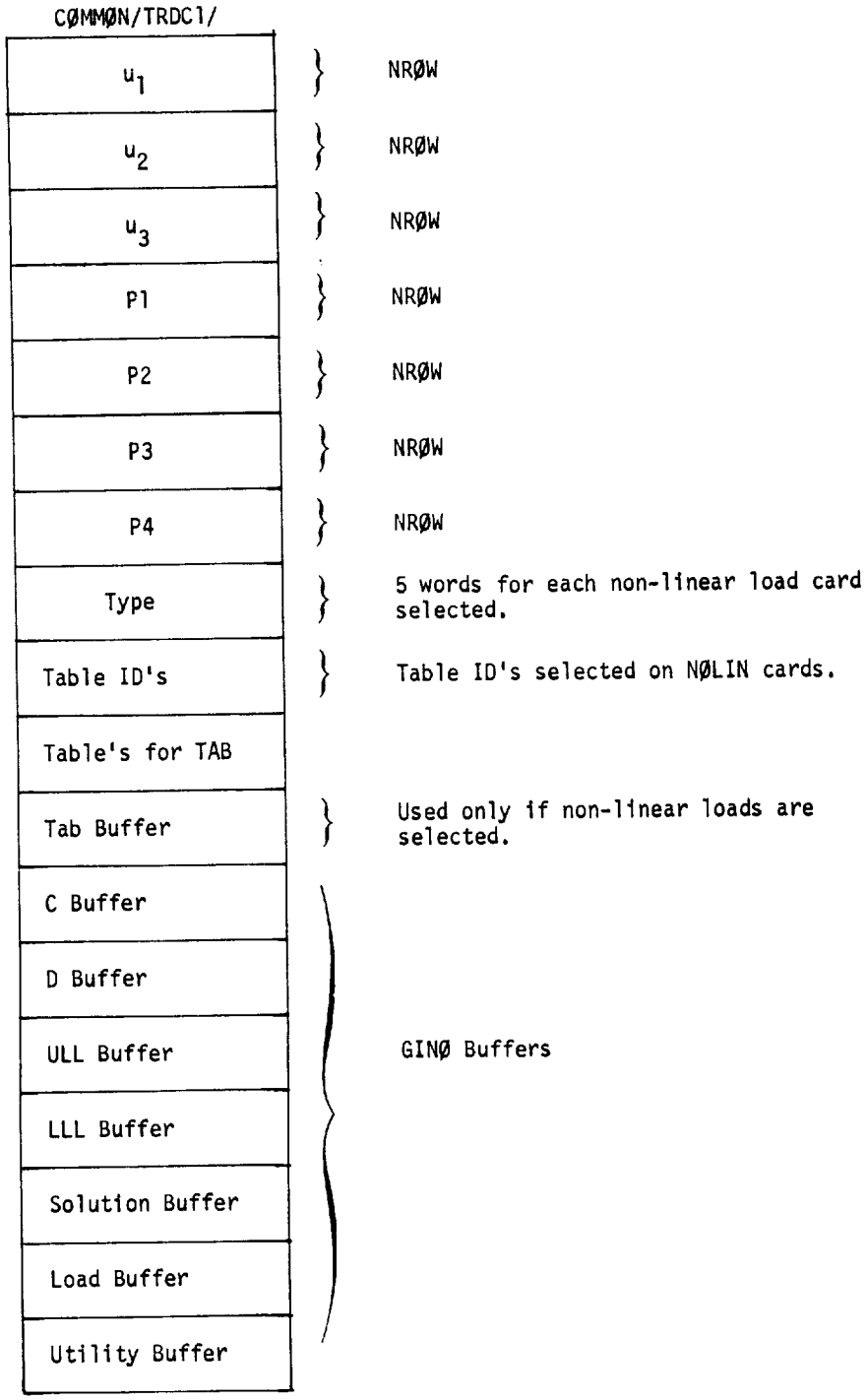
There must be enough core to assemble one time step load.

2. Open Core at /TRD1X/ is illustrated as follows:

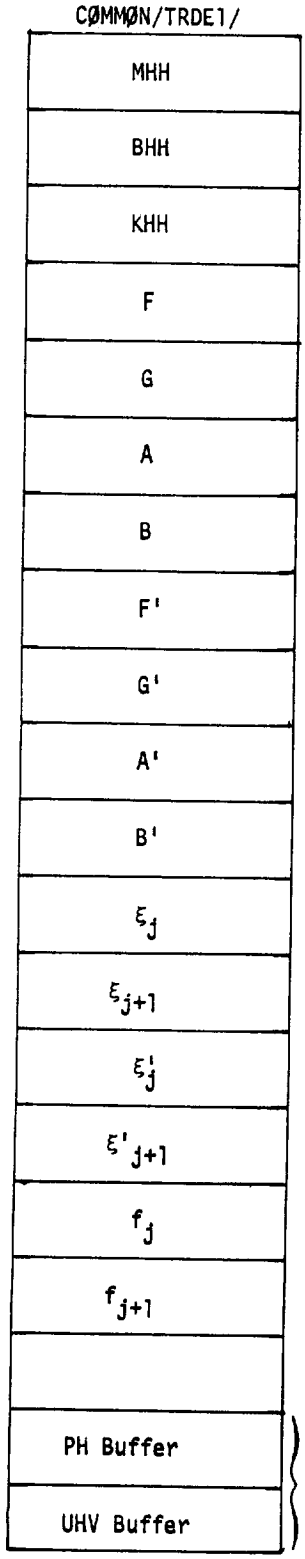


This table is at the bottom of open core through the module.

3. Open Core at /TRDC1/ is illustrated as follows:



4. Open Core at /TRDE1/ is illustrated as follows:



Each section is of length H

2 GINØ buffers

4.65.10 Diagnostic Messages

TRD may issue the following messages

3001, 3002, 3003, 3005, 3007, 3008, 3031, 3044, 3045, 3046.

4.66 FUNCTIONAL MODULE GKAM (GENERAL K ASSEMBLER MODAL)

4.66.1 Entry Point: GKAM

4.66.2 Purpose

To assemble the modal mass, damping and stiffness matrices.

4.66.3 DMAP Calling Sequence

```
GKAM    USETD,PHIA,MI,LAMA,DIT,M2DD,B2DD,K2DD,CASECC/MHH,BHH,KHH,PHIDH/V,N,NØUE/C,Y,
        LMØDES/C,Y,LFREQ/C,Y,HFREQ/V,N,NØM2PP/V,N,NØB2PP/V,N,NØK2PP/V,N,NØNCUP/V,N,
        FMØDE   $
```

4.66.4 Input Data Blocks

USETD - Displacement set definitions table dynamics.

PHIA - Eigenvectors matrix giving the eigenvectors (displacements) in the a set.

MI - Modal mass matrix.

LAMA - Real Eigenvalue Table.

DIT - Direct Input Table.

M2DD - Direct input mass matrix - d set.

B2DD - Direct input damping matrix - d set.

K2DD - Direct input stiffness matrix - d set.

CASECC - Case Control Data Table.

Notes:

1. USETD may be purged if NØUE < 0.
2. PHIA cannot be purged.
3. MI cannot be purged.
4. LAMA cannot be purged.
5. DIT cannot be purged if SDAMP $\neq$ 0 in CASECC.
6. CASECC cannot be purged.
7. M2DD cannot be purged if NØM2PP $\geq$ 0.
8. B2DD cannot be purged if NØB2PP $\geq$ 0.
9. K2DD cannot be purged if NØK2PP $\geq$ 0.

4.66.5 Output Data Blocks

MHH - Modal mass matrix - h set.

BHH - Modal damping matrix - h set.

KHH - Modal stiffness matrix - h set.

PHIDH - Transformation matrix from d set to modal coordinates.

Note: No output matrix can be purged.

4.66.6 Parameters

NØUE - Input-integer-no default. NØUE indicates presence and number of extra points.

LMØDES - Input-integer-default = 0. LMØDES selects the first LMØDES eigenvectors (or all if there are less than LMØDES) to use for the modal coordinates.

LFREQ - Input-real-default = 0.0. If LMØDES = 0, eigenvectors with eigenvalues between LFREQ and HFREQ are used in the modal formulation.

HFREQ - Input-real-default = 0.0. See LFREQ.

NØM2PP - Input-integer-no default. If NØM2PP < 0, M2DD will not be used.

NØB2PP - Input-integer-no default. If NØB2PP < 0, B2DD will not be used.

NØK2PP - Input-integer-no default. If NØK2PP < 0, K2DD will not be used.

NØNCUP - Output-integer-no default. If no direct input matrices exist the problem is considered uncoupled and NØNCUP is set to -1.

FMØDE - Output-integer-no default. The mode number of the first selected eigenvector is stored in FMØDE.

4.66.7 Method

The general system assembly module for the modal method is used when the real eigenvalues for the structure have been determined. With this method, it is possible to decrease the order of the problem without sacrificing accuracy. The module forms the conversion matrix between modal displacements and all free physical displacements of the system. It then forms the general matrices in terms of displacements of the modes and the extra points.

CASECC is read, and the selected structural damping table "id" is stored.

LAMA is read and the selected eigenvalues are stored in core. If an eigenvalue is selected, the corresponding column of PHIA is copied onto PHIDH1, a scratch file.

If extra points are not present (NØUE < 0), PHIDH = PHIDH1. If extra points are present:

$$[\Phi_{dh}] = \left[\begin{array}{c|c} \Phi_a & 0 \\ \hline 0 & I \end{array}\right]. \tag{1}$$

This is accomplished in subroutine GKAM1B.

The "H" matrices are formed:

$$[M_{hh}] = \left[\begin{array}{c|c} m_i & 0 \\ \hline 0 & 0 \end{array}\right] + [\Phi_{dh}]^T [M^2_{dd}] [\Phi_{dh}] , \tag{2}$$

$$[B_{hh}] = \left[\begin{array}{c|c} b_i & 0 \\ \hline 0 & 0 \end{array}\right] + [\Phi_{dh}]^T [B^2_{dd}] [\Phi_{dh}] , \tag{3}$$

$$[K_{hh}] = \left[\begin{array}{c|c} k_i & 0 \\ \hline 0 & 0 \end{array}\right] + [\Phi_{dh}]^T [K^2_{dd}] [\Phi_{dh}] , \tag{4}$$

where m_i = diagonal terms of MI, and

$$b_i = m_i \omega_i g(\omega_i), \tag{5}$$

$$k_i = m_i \omega_i^2 . \tag{6}$$

ω_i is the frequency for the mode from LAMA and $g(\omega_i)$ is the tabular structural damping table selected in CASECC. If no selection is made $g(\omega_i) \equiv 0.0$. The "H" matrices are formed using subroutines GKAM1A, SSG2B, PRETAB, TA3, CALCV, MERGE.

4.66.8 Subroutines

4.66.8.1 Subroutine Name: GKAM1B.

1. Entry Point: GKAM1B.

2. Purpose: To construct $[\Phi_{dh}]$ if extra points are present.

3. Calling Sequence: CALL GKAM1B (USETD,SCR1,SCR2,PHIDH,PHIDH1,MØDES,CØRE,LHSET,NØUE)

USETD - GINØ file number of USETD - integer - input.

SCR1 - GINØ file number of 1st scratch file - integer - input.

SCR2 - GINØ file number of 2nd scratch file - integer - input.

PHIDH - GINØ file number of PHIDH - integer - input.

PHIDH1 - GINØ file number of PHIDH1 - integer - input.

MØDES - Number of modes selected - integer - input.

CØRE - Array of open core.

LHSET - Length of h set - integer - output.

NØUE - Extra point flag NØUE $\geq$ 0 indicates presence of extra points - integer - input.

4.66.8.2 Subroutine Name: GKAM1A.

1. Entry Point: GKAM1A.

2. Purpose: To form $[M_{hh}]$, $[B_{hh}]$, or $[K_{hh}]$.

3. Calling Sequence: CALL GKAM1A (MI,PHIDH,DIT,SCR1,SCR2,IØPT,IHH,NØI2DD,CØRE,MØDES, SDITD,LHSET,I2DD,IMSKIP,SCR3)

MI - GINØ file number of MI - integer - input.

PHIDH - GINØ file number of PHIDH - integer - input.

DIT - GINØ file number of DIT - integer - input.

SCR1 - GINØ file number of scratch 1 - integer - input.

SCR2 - GINØ file number of scratch 2 - integer - input.

SCR3 - GINØ file number of scratch 3 - integer - input.

IHH - GINØ file number of HH file (M, B, or K) being constructed - integer - input.

I2DD - GINØ file number of 2DD file being used with IHH (K2DD, M2DD or B2DD) - integer - input.

IØPT - Flag for equation to use

1 $\Longrightarrow$ MHH

2 $\Longrightarrow$ BHH

3 $\Longrightarrow$ KHH

- integer - input.

NØI2DD - NØI2DD < 0 implies I2DD purged - integer - input.

MØDES - Number of modes selected - integer - input.

SDTID - Id of structural damping table to be used for BHH - integer - input.

LHSET - Length of H set - integer - input.

IMSKIP - Number of records to skip in MI before extracting diagonal terms - integer - input.

CØRE - Array of modes selected.

4.66.9 Design Requirements

Three scratch files are necessary. Open core at /GKAM1X/ is used for mode storage. One packed eigenvector must be held in core.

4.66.10 Diagnostic Messages

Fatal error messages 3007 and 3008 may be issued by GKAM.

4.67 FUNCTIONAL MODULE DDR1 (DYNAMIC DATA RECOVERY - PART 1)

4.67.1 Entry Point: DDR1

4.67.2 Purpose: To transform modal solutions to physical solutions:

$$\{u_d\} = [\Phi_{dh}] \{u_h\} . \qquad (1)$$

4.67.3 DMAP Calling Sequence

```
DDR1   UHV,PHIDH/UDV  $
```

4.67.4 Input Data Blocks

UHV - Solution set displacement vectors.

PHIDH - Transformation matrix from d set to modal coordinates.

4.67.5 Output Data Blocks

UDV - Displacement vectors - d set.

4.67.6 Parameters

None

4.67.7 Method

Subroutine SSG2B is called to compute $\{u_d\}$ as in Equation 1.

4.67.8 Subroutines

DDR1 has no auxiliary subroutines. See section 3.5.13 for a description of SSG2B.

4.67.9 Design Requirements

One scratch file is needed.

4.68 FUNCTIONAL MODULE DDR2 (DYNAMIC DATA RECOVERY - PART 2)

4.68.1 Entry Point: DDR2

4.68.2 Purpose

To compute mode acceleration displacements.

4.68.3 DMAP Calling Sequence

```
DDR2   USETD,UDV,PDF,K2DD,B2DD,MDD,FRL,ULL,LLL,DM/UDV1,UEVF,PAF/V,N,TYPE/V,N,NØUE/V,N,REACT/
       V,N,FRQSET   $
```

4.68.4 Input Data Blocks

USETD - Displacement set definitions table dynamics.

UDV - Displacement vectors - d set.

PDF - Dynamic load matrix for frequency analysis - d set.

K2DD - Direct input stiffness matrix - d set.

B2DD - Direct input damping matrix - d set.

MDD - Dynamic mass matrix - d set.

FRL - Frequency Response List.

ULL - Upper triangular factor of KLL - ℓ set.

LLL - Lower triangular factor of KLL - ℓ set.

DM - Rigid body transformation matrix.

Notes:

1. USETD must not be purged.
2. UDV must not be purged.
3. PDF must not be purged.
4. FRL must not be purged if TYPE = FREQ.
5. MDD must not be purged.
6. ULL, LLL must not be purged.
7. DM must not be purged if REACT $\geq$ 0.

4.68.5 Output Data Blocks

UDV1 - Displacements after mode acceleration - d set.

UEVF - Displacements at the extra points.

PAF - Equivalent load vector for mode acceleration computations - a set.

4.68.6 Parameters

TYPE - Input-BCD-no default. TYPE determines the type of mode acceleration which will be used, TRAN for transient or FREQ for frequency response.

NØUE - Input-integer-no default. NØUE $\geq$ 0 indicates presence of extra points.

REACT - Input-integer-no default. REACT $\geq$ 0 indicates presence of supports.

FRQSET - Input-integer-no default. FRQSET chooses the frequency list if TYPE = FREQ.

4.68.7 Method

The equivalent load vector is computed:

$$\{P_d^a\} = \{P_d\} - [K_{dd}^2]\{u_d\} - [B_{dd}^2]\{\dot{u}_d\} - [M_{dd}]\{\ddot{u}_d\} . \quad (1)$$

For a transient analysis problem $\{u_d\}$, $\{\dot{u}_d\}$, and $\{\ddot{u}_d\}$ are given explicitly. For Frequency Response Analysis:

$$\{\dot{u}_d\} = i\omega\{u_d\} , \quad (2)$$

$$\{\ddot{u}_d\} = -\omega^2\{u_d\}, \quad (3)$$

where ω is the forcing frequency and $\{u_d\}$ is the complex response vector. ω comes from FRQSET in FRL. The vector $\{P_d^a\}$ is the sum of applied loads and inertia loads due to the motion of the system approximated by its lower modes. The static solution using these loads will provide a better answer for displacements.

If extra points are present (NØUE $\geq$ 0), then

$$\{P_d^a\} \Rightarrow \left\{\begin{array}{c} P_a^a \\ \hline P_e \end{array}\right\} , \tag{4}$$

$$\{u_d\} \Rightarrow \left\{\begin{array}{c} u_a^a \\ \hline u_e \end{array}\right\} . \tag{5}$$

$\{u_e\}$ is placed in data block UEVF. Subroutines CALCV and SSG2A perform this calculation.

If supports are present (REACT $\geq$ 0), then

$$\{P_\ell^a\} \Rightarrow \left\{\begin{array}{c} P_\ell \\ \hline P_r \end{array}\right\} , \tag{6}$$

$$\{u_a\} \Rightarrow \left\{\begin{array}{c} u_\ell \\ \hline u_r \end{array}\right\} . \tag{7}$$

Solve for $\{u_a^a\}$:

$$[L_{\ell\ell}]\,[U_{\ell\ell}]\,\{u_\ell^a\} = \{P_\ell^a\} . \tag{8}$$

This is accomplished in subroutine SSG3A.

If supports are present, then

$$\{u_a^a\} = \left\{\begin{array}{c} \{u_\ell^a\} + [D]\,\{u_r\} \\ \hline u_r \end{array}\right\} , \tag{9}$$

otherwise, $\{u_a^a\} = \{u_\ell^a\}$. Subroutine SDR1B performs this calculation.

If extra points are present, then

$$\{u_d^a\} \Longleftarrow \left\{\begin{array}{c} u_a^a \\ \hline u_e \end{array}\right\} . \qquad (10)$$

Note: If the problem type is transient, $\{u_d^a\}$ must be merged with $\{\dot{u}_d\}$ and $\{\ddot{u}_d\}$.

4.68.8 Subroutines Called

CALCV - See section 3.5.5.

SSG2A - See section 3.5.7.

SSG2B - See section 3.5.13.

SSG3A - See section 3.5.18.

SDR1B - See section 3.5.8.

4.68.8.1 Subroutine Name: DDR1A.

1. Entry Point: DDR1A.

2. Purpose: To construct the equivalent load vector $\{P_d^a\}$.

3. Calling Sequence: CALL DDR1A(PDF,K2DD,B2DD,MDD,UDV,PAF,FRL,FRQSET,SCR1,SCR2,SCR3,SCR4, TYPE,SCR5).

PDF, K2DD, B2DD, MDD, UDV, PAF, FRL, SCR1-5 - GINØ file number of appropriate data block - integer - input.

FRQSET - Frequency set list id - integer - input. FRQSET will be used only if TYPE = FREQ.

TYPE - Problem type - BCD - input.

4.68.8.2 Subroutine Name: DDR1B

1. Entry Point: DDR1B.

2. Purpose: To merge displacements with previously computed velocity and acceleration in a transient problem.

3. Calling Sequence: CALL DDR1B (UDV,UAD,UADV).

UDV - GINØ file number of displacement, velocity and acceleration file - integer - input.

UAD - GINØ file number of equivalent displacements - integer - input.

UADV - GINØ file number of new displacements, velocity and acceleration - integer - input.

4.68.9 Design Requirements

Open core for DDR2 begins at /DDR1X/. Open core for DDR1A begins /DDRA1/. Open core for DDR1B begins /DDRB1/. Six scratch files are needed.

4.68.10 Diagnostic Messages

None.

4.69 OUTPUT MODULE XYPLØT (X-Y DATA PLOTTER)

4.69.1 Entry Point: XYPLØT

4.69.2 Purpose

To process information supplied by module XYTRAN through a single data block and output to either PLT1 (BCD plot tape) or PLT2 (binary plot tape) for labeling and plotting X-Y data on an off-line plotter.

4.69.3 DMAP Calling Sequence

```
XYPLØT     XYPLTT//  $
```

4.69.4 Input Data Blocks

XYPLTT - Plotting Control Values Table. Note if XYPLTT is purged, XYPLØT returns control without action.

4.69.5 Output Data Blocks

None. (All output consists of physical tapes produced for off-line plotters and possibly user warning messages to the installation output unit for printing).

4.69.6 Parameters

None.

4.69.7 Method

XYPLØT initially determines open core size and assigns buffers for its input file and output file. The remaining core is used to store data points read in for each plot. The input file is then opened and spaced forward over the header record containing the data block name. Should the system not be able to locate this file, a warning message is output and XYPLØT returns control to the calling program without further action. Otherwise XYPLØT reads in the first I.D. record from the input file. A check is made to determine if the word count of this record is correct. If not, the following records are checked until either the correct

word count is found or the error count reaches a specified limit. If the specified limit is reached, XYPLØT assumes the input file is invalid and returns control to the calling program after printing a warning message.

If the I.D. record had the proper word count, XYPLØT checks if new axes are necessary. If not, the next data record is read, and the data pairs are plotted on the previous axes. When new axes are necessary, a check is made to determine if they go on the lower half of a plot. If not, XYPLØT makes a number of I.D. data validity checks. Whenever possible, where I.D. data are questionable, default values are assigned and processing continues following a warning message that this particular plot may be invalid.

After the validity checks, XYPLØT terminates the previous plot and initializes the plotting parameters for the NASTRAN plotting software. This is done for each new plot so that it is possible to produce alternate plots on two different plotters. Normally, however, plots will be done for only one plotter on any single entry to XYPLØT. If required, a new plot is initiated, and curve and axes titles are prepared from the I.D. data and generated. If not a new plot, only the axes titles are done.

At this time XYPLØT computes the constants which will be used to transform the curve data into actual plotter counts. These constants are saved and used until new axes are drawn.

Following this, XYPLØT determines if any tick marks are to be placed along the X axis and at the X maximum and X minimum lines. If there are to be tick marks, the number and spacing (linear or logarithmic) is computed for them and plotted. As the X direction tick marks are prepared, a check is made to determine if Y grid lines are requested. If so, a grid line is prepared at each tick mark and plotted. Logarithmic tick mark labels are prepared and plotted at the same time as the tick marks and X grid lines, if any.

After the tick marks are completed, the X and Y axes are plotted if requested. Finally the linear tick marks are labeled in both the X and Y directions if requested.

Once the curve titles, tick marks, and labeling have been accomplished, XYPLØT reads in the next record from the input data file. Normally all the data pairs for any I.D. record can be brought into core memory with a single read. However, provision is made for additional reads if the open core space is not sufficient to contain all the data on the initial read. A check is made to determine if there are an even number of data values (i.e., an X and Y value for each data

point). If not, a warning message is printed and the last value ignored. The data are then checked against the previously defined X and Y minimums and maximums. Any data outside these limits are ignored and not plotted. The remaining data points are then converted to plotter counts and plotted in one of three modes. The three modes are: point plot with choice of symbol; line plot; combination of the first two.

After finishing the data, XYPLØT reads in the next I.D. record and continues as before until an end-of-file is reached. At this point it closes the input file, terminates the current plot and returns control to the calling program.

4.69.8 Subroutines

XYPLØT calls the following plotter utility subroutines: AXIS, LINE, PRINT, SØPEN, PLTSET, STPLØT, SYMBØL, TIPE, TYPFLT, and TYPINT. The descriptions of these subroutines may be found in section 3.4.

4.69.9 Design Requirements

4.69.9.1 Allocation of Core Storage

XYPLØT uses open core for two GINØ buffers and the remainder as one large buffer for data points. It appears as follows:

Normally the data pairs buffer will be sufficiently large to hold all the data pairs for a single curve at one time. However, this is not necessary and XYPLØT could operate if the data pairs buffer were only two words long, although not efficiently. As an output module, XYPLØT has been programmed to avoid any system fatal errors. The worst condition that should occur is that no plots are produced. In all cases XYPLØT returns to the calling program so that other system functions may be continued.

4.69.9.2 Environment

The beginning of open core for XYPLØT is defined by /XYPLXX/. XYPLØT uses no scratch files. Common storage requirements consist of /XXPARM/ and /PLTDAT/ which are defined in the block data deck PLØTBD which must be loaded with XYPLØT. /CHAR94/ and SYMBLS/ are also defined in PLØTBD and are necessary for the subroutines called by XYPLØT. See section 2.5 for a description of these common blocks.

When XYPLØT is called, there must be at least one physical tape set up to receive the plotted output, otherwise XYPLØT returns to the calling program without further action.

4.69.10 Diagnostic Messages

Diagnostic messages 991 through 997 may be output on the installation printer device as a result of XYPLØT operation. Generally they are self-explanatory and usually point out particular plots which are questionable rather than giving the user a precise method of solving the problem. This is not possible since XYPLØT receives all its information through a series of other modules rather than from the user directly. See section 6 of the User's Manual for details.

4.70 OUTPUT MODULE ØFP (OUTPUT FILE PROCESSOR)

4.70.1 Entry Point: ØFP

4.70.2 Purpose

ØFP outputs to the system output file, in user-oriented, self-explanatory formats, data blocks prepared for output by other functional modules.

4.70.3 DMAP Calling Sequence

```
ØFP  DB1,DB2,DB3,DB4,DB5,DB6//V,N,CARDNØ  $
```

4.70.4 Input Data Blocks

One to six input data blocks in the output order desired. Any or all input data blocks may be purged.

4.70.5 Output Data Blocks

None

4.70.6 Parameters

CARDNØ - Input and output - integer - default = 0. CARDNØ is incremented by one and punched in columns 73-80 for each card punched by ØFP.

4.70.7 Method

4.70.7.1 Overall Logic Flow

The ØFP logic consists of defining one GINØ buffer and then entering one overall loop of six passes (one pass for each data block). All input data blocks are then handled identically one at a time.

Within each data block, each odd numbered (Identification) record and its respective immediately following even numbered (Data) record are considered as a pair, and is a completely separate entity. There is, and need be, no correspondence between these two records and the previous two records, or between these two records and the following two records.

Thus, within the loop for a given data block, after the file on which the data block resides is opened and its header record is skipped, ØFP reads an Identification record, defines various pointers and descriptors, and then, if any data are present in the Data record, processes this data line by line until the end-of-record is reached. This process continues for all Identification-Data record pairs.

4.70.7.2 Defining Descriptors and Pointers

Because ØFP was confronted with outputting a vast array of data classes having many data format and heading format configurations, it was decided that in order to keep ØFP from becoming a mammoth module of format statements, a system of pointers would be used in conjunction with all the different micro-format elements required.

Information in the Identification record is sufficient to select an initial class pointer. This class pointer, with the addition of a subclass pointer, points to an array of six pointers, five of which define five micro-line formats (from the master set of micro-line formats), and one of which points to a string of micro-data format pointers. These micro-data format pointers then each point to a micro-data format capable of outputting a single variable.

This design is such as to make possible the definition of macro-formats and to allow for easy modification and addition of more output data classes.

4.70.8 Subroutines

4.70.8.1 Subroutine Name: ØFPPUN

1. Entry Point: ØFPPUN
2. Purpose: To write output on the system punch unit.
3. Calling Sequence: CALL ØFPPUN (BUF,NWDS,IØPT,IDD,PNCHED)

   BUF - Array to be output.

   NWDS - Number of words in BUF to output.

   IØPT - { 1 = Vector output.
          { 2 = General output.

   IDD - { 0 = SØRT1 (1st word Integer).
         { 1 = SØRT2 (1st word Real).

PNCHED - {.FALSE. = Punch heading cards.
{.TRUE. = Do not punch heading cards.

4.70.8.2 Subroutine Name: ØFP1

1. Entry Point: ØFP1
2. Purpose: To call PAGE and write five micro-line formats.
3. Calling Sequence: CALL ØFP1

4.70.8.3 Subroutine Name: ØFP1A

1. Entry Point: ØFP1A
2. Purpose: An auxiliary routine to ØFP1. Called by ØFP1 only.
3. Calling Sequence: CALL ØFP1A(LINE)

   LINE - Integer - Branch to format pointer.

4.70.8.4 Block Data Subprogram Name: ØFP1BD

ØFP1BD defines common block /ØFPBD1/.

4.70.8.5 Block Data Subprogram Name: ØFP2BD

ØFP2BD defines common block /ØFPBD2/.

4.70.8.6 Block Data Subprogram Name: ØFP3BD

ØFP3BD defines common block /ØFPBD3/.

4.70.8.7 Block Data Subprogram Name: ØFP4BD

ØFP4BD defines common block /ØFPBD4/.

4.70.8.8 Block Data Subprogram Name: ØFP5BD

ØFP5BD defines common block /ØFPBD5/.

4.70.9 Design Requirements

The common blocks listed above interface between the main subroutines ØFP and ØFP1A. In addition CØMMØN/ØFPXXX/ is used to define open-core which contains the following.

L1, L2, L3, L4, L5 - Five words which indicate the five format numbers defining the heading for the current data being output.

ID - A fifty word buffer for storage of the first fifty words of an identification record from the data block to be output.

BUFF - A GINØ buffer.

The pointer system required by the design operates as described below. The arrays B,C,D,E, and ESINGL, referenced in the discussion below appear in subroutine ØFP.

1. The variable I is set equal to the Data type specified in the data block. The variable J is set equal to the class of data: 1 = Real - SØRT1, 2 = Complex - SØRT1, etc. Then the base pointer, CPØINT = B(I,J), is found.

B array

	SØRT 1		SØRT 2	
	Real	Complex	Real	Complex
Data type 1	0	130	120	132
Data type 2	2	134	122	136
.	4	138	124	140
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Data type N	.	.	.	.
	For Future Expansion			

Example:
For I = 2, J = 4.

CPØINT = B(I,J)
= 136

2. CPØINT is a index into the C array. Define: DPØINT = C(CPØINT). DPØINT is an index into the D array. Also

L1 = C(CPØINT + 1),

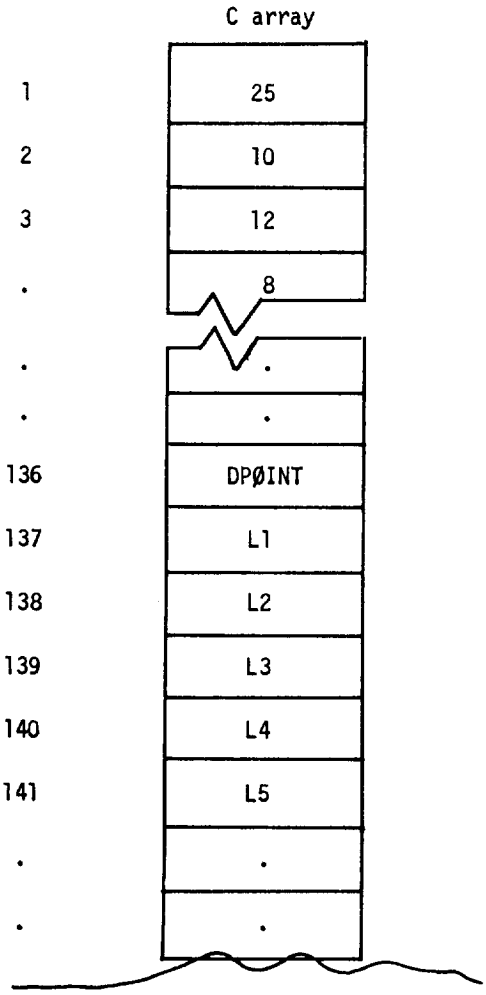
L2 = C(CPØINT + 2),

L3 = C(CPØINT + 3),

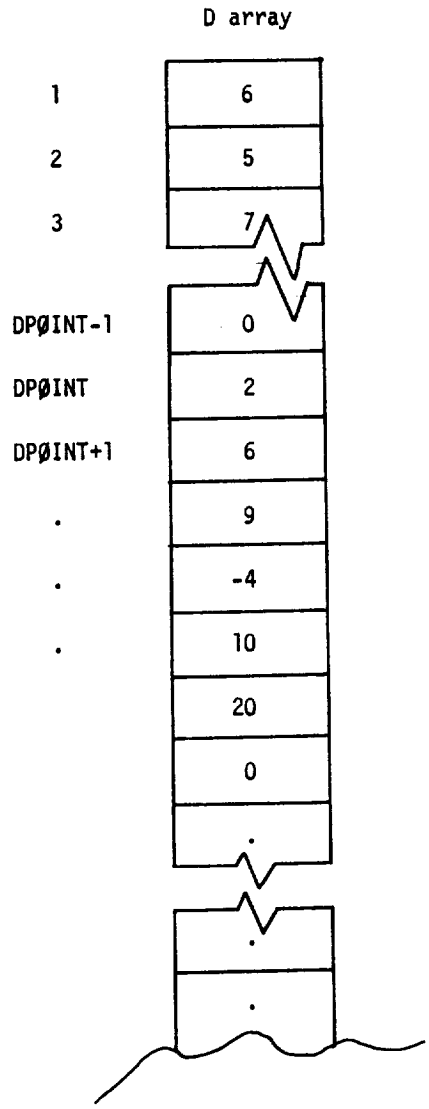
L4 = C(CPØINT + 4),

and L5 = C(CPØINT + 5).

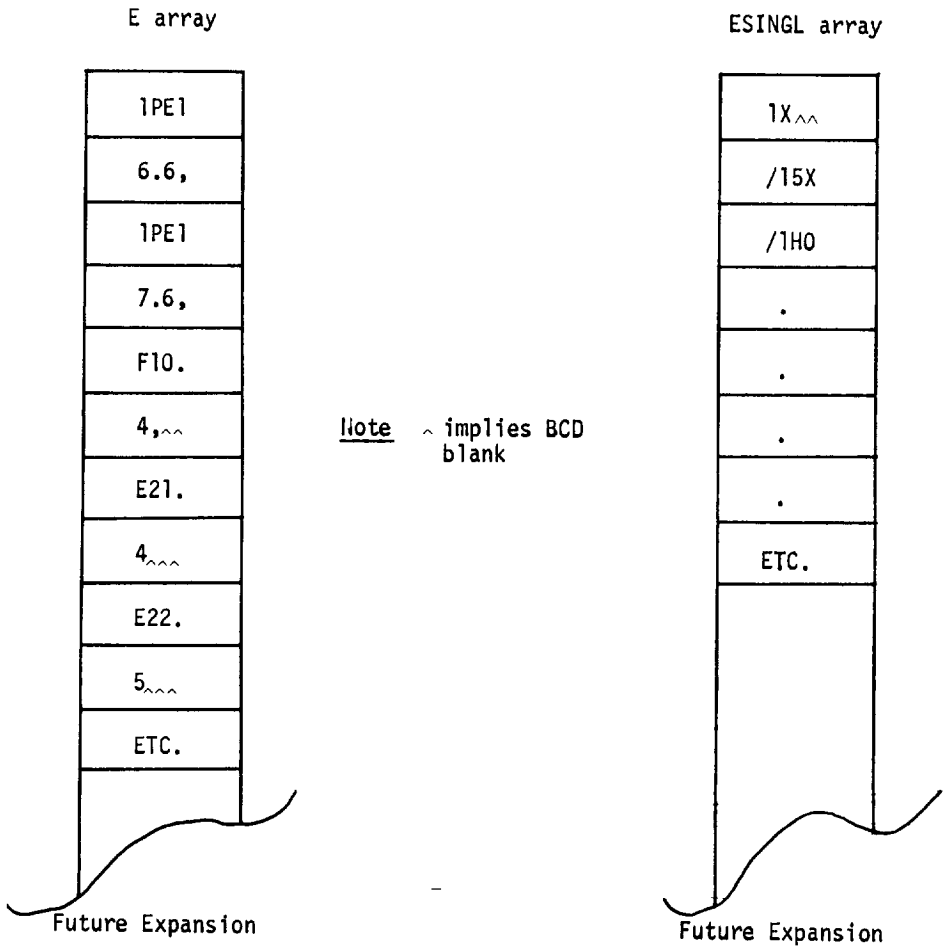
These are the 5 line format numbers which make up the heading format for the type of data currently being processed.



3. Word DPØINT of the D array defines the beginning of a string of pointers into the E array. This string is terminated by the first word containing a zero. Each word of this string thus defines a string of words in the E array which contains Hollerith data for construction of a format. Should a word in the D array be negative, the absolute value is used to point into the ESINGL array which contains Hollerith data also.



4. The E array contains Hollerith data pertaining to the output of a variable; the ESINGL array contains Hollerith data pertaining to spacing and carriage control only.



4.70.10 Diagnostic Messages

If, during some phase of outputting a data block, ØFP encounters an error condition, work on that data block will cease, a warning message will be printed, and a call to the NASTRAN table-printer for table printing of this data block will be made. ØFP will then continue processing the remaining input data blocks.

4.71 OUTPUT MODULE MATPRN (GENERAL MATRIX PRINTER)

4.71.1 Entry Point: MATPRN.

4.71.2 Purpose

To print general matrix data blocks.

4.71.3 DMAP Calling Sequence

```
MATPRN    KGG,PL,PG,B2PP,UPV//  $
```

4.71.4 Input Data Blocks

KGG - Any matrix data block.

PL - Any matrix data block.

PG - Any matrix data block.

B2PP - Any matrix data block.

UPV - Any matrix data block.

Notes:

1. Any or all input data blocks can be purged.
2. If any data block is not a matrix, the TABPT routine will be called.

4.71.5 Output

The non-zero band of each column of the input matrix is unpacked and is printed in single precision format on the system output file.

4.71.6 Parameters

None.

4.71.7 Method

Subroutine MATDUM is called for each non-purged input file.

4.71.8 Subroutines

MATDUM - See subroutine description, section 3.4.28.

4.72 OUTPUT MODULE MATGPR (DISPLACEMENT METHOD MATRIX PRINTER)

4.72.1 Entry Point: MATGPR

4.72.2 Purpose

To print displacement method matrices, identifying values with external grid point numbers.

4.72.3 DMAP Calling Sequence

```
MATGPR    GPL,USET,SIL,ANYMAT//C,N,CØLSET/C,N,RØWSET    $
```

4.72.4 Input Data Blocks

GPL - Grid Point List (This may also be GPLD if extra points are present.)

USET - Displacement set definitions table (This may also be USETD if extra points are present.)

SIL - Scalar Index List (This may also be SILD if extra points are present.)

ANYMAT- Any displacement method matrix.

Notes:

1. Unless CØLSET = RØWSET = 'H', GPL, USET and SIL must be present.
2. If ANYMAT is purged, MATGPR will return.

4.72.5 Output Data Blocks

The non-zero terms of ANYMAT are given external identification and printed on the system output file.

4.72.6 Parameters

CØLSET - Input-BCD-no default. CØLSET indicates the set to which the columns of ANYMAT belong. If CØLSET is not one of the following: M,Ø,R,SG,SB,L,A,F,S,N,G,E,P,NE,FE,D,H then MATGPR will return.

RØWSET - Input-BCD-default = X. RØWSET indicates the set to which the rows of ANYMAT belong. If RØWSET is not a legal set name, RØWSET = CØLSET.

4.72.7 Method

The BCD parameters CØLSET and RØWSET are converted to bit positions in USET. CØLSET must be one of the following 17 symbols: M,Ø,R,SG,SB,L,A,F,S,N,G,E,P,NE,FE,D,H or else MATGPR will return. If ROWSET is not a legitimate symbol RØWSET = CØLSET.

GPL, USET, and SIL are placed in core. Each column and non-zero row element is identified according to the following scheme:

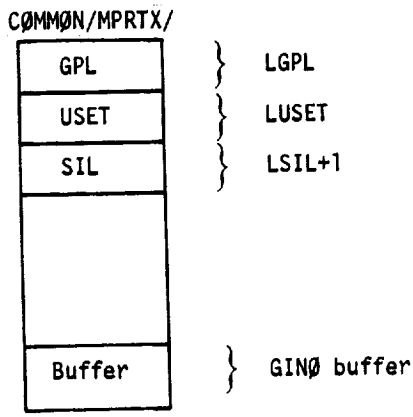
1. USET is searched for the number of members belonging to the g set (p set if USETD is used) before the current member of the matrix set.
2. This number is looked up in SIL to obtain the internal grid point number and type of point (scalar, grid, or extra).
3. The internal grid point number points into GPL for the external ID.

4.72.8 Subroutines

MATGPR has no auxiliary subroutines.

4.72.9 Design Requirements

1. Open core is defined at /MPRTX/.
2. Layout of open core is as follows:



4.72.10 Diagnostic Messages

MATGPR may issue the following diagnostic messages:

3007 and 3008.

4.73 OUTPUT MODULE MATPRT (MATRIX PRINTER)

4.73.1 Entry Point: PRTINT

4.73.2 Purpose

To print a matrix data block.

4.73.3 DMAP Calling Sequence

```
MATPRT    X//C,N,RC/C,N,Y  $
```

4.73.4 Input Data Block

X - Matrix data block to be printed. If X is purged, then nothing is done.

4.73.5 Output Data Blocks

None.

4.73.6 Parameters

RC - Indicates whether X is stored by rows (RC = 1) or columns (RC = 0) - integer - input.

Y - Indicates whether X is to be printed (Y < 0, do not print X; Y > 0, print X) - integer - input - default value = -1.

4.73.7 Method

Each column (or row) of the matrix is broken into groups of 6 terms (3 terms if complex) per printed line. If all the terms in a group = 0, the line is not printed. If the entire column (or row) = 0, it is not printed. If the entire matrix = 0, it is not printed.

4.73.8 Subroutines

4.73.8.1 Subroutine Name: INTPRT

1. Entry Point: INTPRT
2. Purpose: To print a matrix data block using subroutine MATPRT.
3. Calling Sequence: CALL INTPRT (A,CR,Ø,NAME)

where:

A - Storage for 1 column (row) of the matrix + 1 GINØ buffer.

CR - { 0 if the matrix is stored by columns.
     { 1 if the matrix is stored by rows.

Ø - { 0 if the matrix is not to be printed.
    { 1 if the matrix is to be printed.

NAME - 8 character name of the matrix (2 words, 4 characters per word).

3. Method: Subroutine MATPRT is called to print the matrix. Whenever MATPRT returns for a matrix name or column/row id to be printed, the name of the matrix ($NAME_1$,$NAME_2$) or the column or row id (as indicated by 'CR'), is printed.

4.73.8.2 Subroutine Name: MATPRT

1. Entry Points: MATPRT, PRTMAT

2. Purpose: To print a matrix data block.

3. Calling Sequence: CALL MATPRT ($\$N_1$,$\$N_2$,A,ØPT,CØLNUM)

CALL PRTMAT ($\$N_1$,$\$N_2$)

CØMMØN/XXMPRT/MCB(7)

where:

N_1 - FØRTRAN statement number defining the return executed whenever a new page has been started (the calling program is expected to print the matrix and column id. CØLNUM = current column number).

N_2 - FØRTRAN statement number defining the return executed whenever the column id must be printed in the middle of a page (CØLNUM = current column number).

A - Storage for 1 column of the matrix + one GINØ buffer.

ØPT - See subroutine description for VECPRT, below, for the explanation of this argument.

CØLNUM - Current column number being printed (output).

MCB - Matrix control block.

3. Method: The matrix is unpacked and printed one column at a time. Whenever either of the nonstandard returns ($\$N_1$,$\$N_2$) is executed, the calling program must call PRTMAT to continue the printing of the matrix.

4. Additional Subroutines Called: VECPRT.

4.73.8.3 Subroutine Name: VECPRT

1. Entry Points: VECPRT, PRTVEC

2. Purpose: To print a vector.

3. Calling Sequence: CALL VECPRT ($\$N_1$,$\$N_2$,P,N,A,ØPT)

CALL PRTVEC ($\$N_1$,$\$N_2$)

where:

N_1 - FØRTRAN statement number defining the return executed whenever a new page has been started (the calling program is expected to print the vector id and any other subtitles desired).

N_2 - FØRTRAN statement number defining the return executed whenever the vector id is to be printed in the middle of a page.

P - Vector type and precision.

N - Number of components in the vector.

A - Location of the vector.

ØPT -
- = 0 if all the vector components are to be printed, regardless of its sparsity, and if it is to be printed starting on a new page if it will not fit on the current page.
- = +1 if only the printed lines which would have at least one non-zero component are to be printed, and if the vector is to be printed starting on a new page if it will not fit on the current page.
- = -1 if only the printed lines which would have at least one non-zero component are to be printed, and if as much of the vector as possible is to be printed on the current page.

3. Method: The vector will be printed as a single precision real or complex vector. The components will be printed 6 per line if real, 3 per line if complex. In addition, the first and last components of each line will be identified on each side of the line by their respective component members. In addition whenever either of the nonstandard

returns (N_1,N_2) is executed the calling program must call PRTVEC to continue the printing of the vector.

4. Additional Subroutines Called: FØRMAT.

4.73.8.4 Subroutine Name: FØRMAT

1. Entry Point: FØRMAT

2. Purpose: To print a line of 1 to 6 real numbers (optionally centered) preceded and followed by integer id's of the first and last number printed.

3. Calling Sequence: CALL FØRMAT (A,N1,N2,N3,L1,L2)

where:

A - Array from which the 1 to 6 real numbers are to be printed.

N1 - Index of the 1st number in the array to be printed.

N2 - Index of the last number in the array to be printed.

N3 - Increment to be used in extracting the 2nd, 3rd, etc., numbers in the array to be printed.

L1 - Integer id of the 1st number to be printed.

L2 - Integer id of the last number to be printed.

3. Method: If L1 and L2 are both positive, the numbers will be centered on the page. If either L1 or L2 is not positive, the numbers will be printed based upon the centering of 6 numbers.

4.73.9 Design Requirements

Open core is defined at /XXPRTI/. Open core contains one GINØ buffer followed by one unpacked real or complex single precision column of the matrix.

4.74 OUTPUT MODULE SEEMAT (PICTORIAL MATRIX PRINTER)

4.74.1 Entry Point: SEEMAT

4.74.2 Purpose

To show nonzero matrix elements on printer or plotter output positioned pictorially by row and column within the outlines of the matrix.

4.74.3 DMAP Calling Sequence

SEEMAT M1,M2,M3,M4,M5//C,N,{PRINT / PLØT}/V,N,PFILE/V,N,PACK/C,N,PLØTTER/C,N,MØDELN1/C,N,
MØDELB1/C,N,MØDELN2/C,N,MØDELB2 $

Note that parameters PLØTTER, MØDELN1, MØDELB1, MØDELN2 and MØDELB2 are all described in paragraph 4 in section 4.74.6.

4.74.4 Input Data Blocks

M1, M2, M3, M4, M5 — Matrix Data Blocks, any of which may be purged.

4.74.5 Output Data Blocks

None. The formatted matrix picture is output on the system output file or on a plot tape depending on the value of the first parameter.

4.74.6 Parameters

1. PRINT implies use of the system output file. (Any value other than PLØT implies PRINT). PLØT implies use of one of the plotters. Either of the plotter tapes PLT1 or PLT2 will be used, depending on the type of plotter requested.

2. PFILE is the plot number (Used only if first parameter is PLØT).

   Input/output variable integer parameter. Frame or sheet number. The value of this parameter will be incremented by one (1) for each frame (sheet) created by SEEMAT.

   The default value for this parameter is 0.

3. PACK is reserved for a future modification that will allow the representation of a nonzero block of the matrix with a single character. This parameter may be specified as C,N only. (see example in paragraph 4 below)

4. Plotter Name and Model Identification (Used only if parameter 1 = PLØT.)

Each plotter name has associated with it two model identifiers. Each of these model identifiers may either be an integer (MØDELNi) value or BCD (MØDELBi) value. If model identifier "i" (i = 1, 2) is an integer, insert its value for MØDELNi; if model identifier "i" (i = 1, 2) is BCD, insert its value for MØDELBi. In either case, specify the other value for model identifier "i" (MØDELBi and MØDELNi, respectively) as C,N only.

Below is a list of model identifiers allowable for each plotter name. A detailed explanation of this list can be found in section 4 of the User's Manual. Each plotter has associated with it a default model and several optional models. The model underlined is the default model. To access this default model, do not specify any of the last four DMAP parameters. For example to specify the CALCØMP 765, 205 (see section 4 of the User's Manual) the following DMAP statement may be used:

```
SEEMAT     M1,M2,M3,M4,M5//C,N,PLØT/V,N,PFILE/C,N/C,N,CALCØMP  $
```

Plotter Name	Model Identifiers
BL	LTE,30 / STE,30
EAI	3500,300 / 3500,45
SC	4020,0
CALCØMP	765,205 / 765,210 / 765,105 / 765,110 / 763,205 / 763,210 / 763,105 / 763,110 / 565,205 / 565,210 / 565,105 / 565,110 / 565,305 / 565,310 / 563,205 / 563,210

Plotter Name	Model Identifiers
	563,105 563,110 563,305 563,310
DD	80,B
NASTRAN	M,O T,O D,O M,1 T,1 D,1

where:

BL is a Benson-Lehner plotter

EAI is an Electronic Associates plotter

SC is a Stromberg Carlson plotter

CALCØMP is a California Computer plotter

DD is a Data Display plotter

NASTRAN is the NASTRAN general purpose plotter

4.74.7 Method

The matrix is partitioned into blocks which can be printed on a single sheet of output paper or plotted on a single frame or sheet of plotter output media. Only blocks containing nonzero elements will be printed. Row and column indices are indicated. The user of this module is cautioned to make sure that his line count limit is large enough. A default of 20,000 lines is provided by NASTRAN. This may be changed via the statement "MAXLINES = value" in the NASTRAN Case Control Deck. The transpose of the matrix is always printed.

The columns of the matrix are examined for nonzero terms. Let the matrix be partitioned into blocks, where a block consists of NL columns and 100 rows, where NL is the number of data lines per page obtained from /SYSTEM/. For each block containing nonzero terms, a BCD block image is stored in open core in packed bit format. Only blocks containing nonzero terms are stored. When NL columns have been passed, the blocks containing nonzero terms are printed on

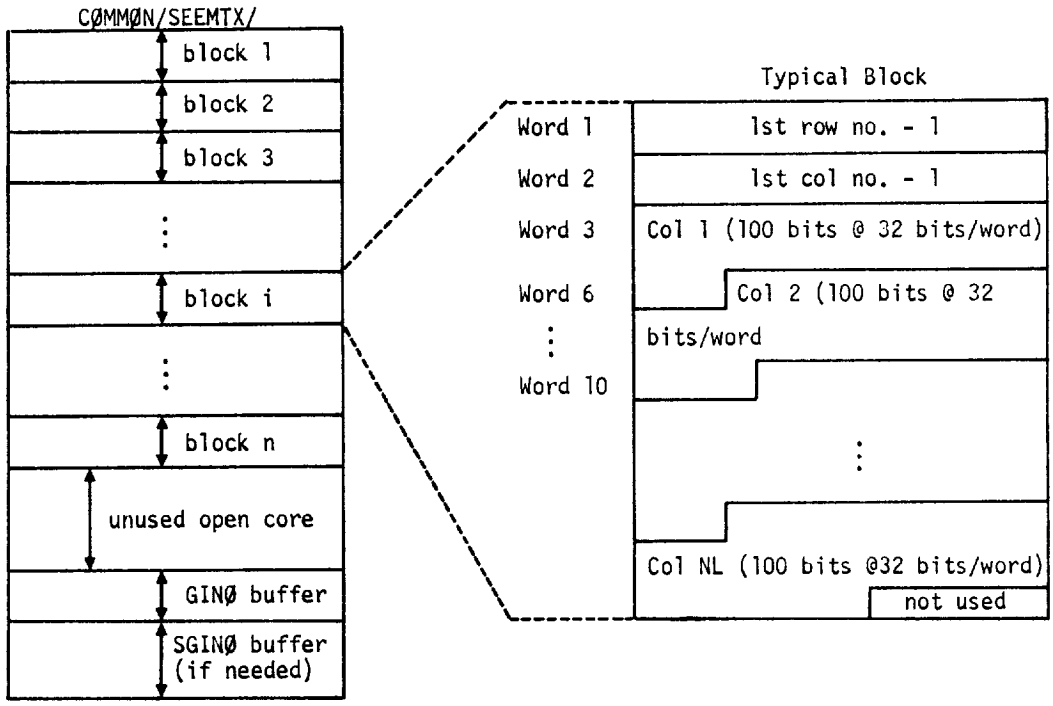
the system output file or plotted. Note that since NASTRAN matrices are stored by column, the transpose of the matrix is what actually appears on the printed or plotted output. Blocks used for the first NL columns may now be re-used for subsequent groups of NL columns. This process i continued until all columns of the matrix have been processed. As many as five matrices may be handled during a single call to SEEMAT.

4.74.8 Subroutines

The plotter environment subroutines are utilized by SEEMAT. See section 3.4 for descriptions of the plotter utility routines.

4.74.9 Design Requirements

4.74.9.1 Open Core Design



4.74.9.2 Data Requirements and Restrictions

1. All nonpurged input data blocks must be matrices. Error diagnostics will occur in the unpacking routines if an attempt is made to input a table data block to SEEMAT.

2. If the number of blocks needed overflows the available open core (e.g., a large full matrix can do this), a nonfatal diagnostic message will be output on the System Output File and processing for that matrix will be terminated. The user may decrease NL by adding a Case Control Card LINE = NL as a means of overcoming this restriction (printer only). Since the type of matrix for which one is interested in seeing the topology is usually sparse and at least partially banded, this restriction should not prove serious.

4.74.10 Diagnostic Messages

Diagnostic conditions detected by SEEMAT are nonfatal and result in appropriate error messages and termination of the processing of the current input matrix data block. The one exception is the condition of no open core for GINØ buffers, which should not occur in practice.

4.75 OUTPUT MODULE TABPT (TABLE PRINTER)

4.75.1 Entry Point: TABPT.

4.75.2 Purpose

To print table data blocks.

4.75.3 DMAP Calling Sequence

```
TABPT    TAB1,TAB2,TAB3,TAB4,TAB5//   $
```

4.75.4 Input Data Blocks

TAB1 - Any NASTRAN data block.

TAB2 - Any NASTRAN data block.

TAB3 - Any NASTRAN data block.

TAB4 - Any NASTRAN data block.

TAB5 - Any NASTRAN data block.

Note: Any or all input data blocks can be purged.

4.75.5 Output

Each word in a data block is identified as real, BCD, or integer and printed on the system output file. The trailer data is also printed.

4.75.6 Parameters

None.

4.75.7 Method

Subroutine TABPRT is called for each non-purged input file.

4.75.8 Subroutines

TABPRT - See subroutine description, section 3.4.29.

4.76 OUTPUT MODULE PRTMSG (MESSAGE WRITER)

4.76.1 Entry Point: PRTMSG

4.76.2 Purpose

To process a data block of user-oriented messages.

4.76.3 DMAP Calling Sequence

```
PRTMSG     MSG//  $
```

4.76.4 Input Data Blocks

MSG - Messages to be printed (if purged, nothing is done).

4.76.5 Output Data Blocks

None

4.76.6 Parameters

None

4.76.7 Method

In addition to messages, the MSG data block may contain titles and subtitles. Before the first message is printed, a new page is started. From then on, a message count is maintained so as to start another new page when the maximum number of lines per page is exceeded. All messages are assumed to be only one line long. However, there is logic included to provide for messages of more than one line, forcing a new page at any time, and the alteration of titles and subtitles at any time. The description below of subroutine WRTMSG details all the included logic capability.

4.76.8 Subroutines

PRTMSG uses the utility routines EJECT and FREAD (see sections 3.4.62 and 3.4.15).

4.76.8.1 Subroutine Name: PRTMSG

1. Entry Point: PRTMSG

2. Purpose: To print the user messages in the MSG data block.

3. Calling Sequence: CALL PRTMSG

CØMMØN/ØUTPUT/TITLE(32,6) - See ØUTPUT miscellaneous table description in section 2.5.

Where:

TITLE = NASTRAN title, subtitle, label, and three extra subtitles.

4. Method: Open the MSG data block, and skip record 0. If the MSG data block does not exist, nothing else is attempted. Otherwise, the three extra subtitles are set to all blanks, and WRTMSG is called.

4.76.8.2 Subroutine Name: WRTMSG (General purpose subroutine)

1. Entry Point: WRTMSG

2. Purpose: To process a data block of user-oriented messages.

3. Calling Sequence: CALL WRTMSG (MSG)

CØMMØN/SYSTEM/ - See SYSTEM table description in section 2.4.1.8.

Where:

MSG = GINØ file name of the MSG data block.

and in /SYSTEM/

MAXLIN = Maximum number of lines permitted per page.

CØUNT = Number of lines thus far printed on the current page.

4. Method:

a. Save the current NASTRAN title, subtitle, and label. Force the first message to start on a new page (CØUNT=MAXLIN).

b. Read one word (N) from MSG. If an end-of-record condition occurs, force the first message in the next record to start on a new page (CØUNT=MAXLIN). Then repeat this step.

c. If N < 0, the next 32 words are assumed to be a replacement for TITLE (1-32,N). Force the next message to start on a new page (CØUNT=MAXLIN). Repeat step b.

d. If N > 0, a list and format follow. The next N items are assumed to be the list items. If N = 0, only a format follows.

e. Read one word (NF) from MSG. If NF < 0, NF = number of lines to be generated by this message (repeat this step). If NF = 0, this message will be printed starting on a new page. (CØUNT=MAXLIN, repeat this step). Unless otherwise instructed, this subroutine assumes that each message will generate only 1 line of output. In either case, integer function EJECT is called to maintain the page line count. If this message will not fit on this page, any extra title(s) explicitly specified are printed below the NASTRAN title, subtitle, and label.

f. If NF > 0, NF = size of the format (the format must be a continuous string of characters, contrary to the usual NASTRAN method of specifying at most 4 characters per word). The next NF words are assumed to be the format to be used with the list items read in step d, in one the following FØRTRAN statements:

```
      WRITE (MØ,FØR)  [if no list is specified].
or
      WRITE (MØ,FØR)  (LIST(I), I = 1,N)
```

Repeat step b.

g. When the end of the MSG data block is encountered in step b, the NASTRAN title, subtitle, and label are restored, and the MSG data block is closed with a rewind.

5. Design Requirements:

a. The message data block (MSG) must be opened before calling WRTMSG.

b. In general, a set of messages is one record of the data block. Each set of messages will start on a new page.

4.76.9 Design Requirements

Open core is defined at /XXPMSG/, and is used only for one GINØ buffer which is defined at the beginning of open core.

4.77 OUTPUT MODULE PRTPARM (PARAMETER AND DMAP MESSAGE PRINTER)

4.77.1 Entry Point: PRTPRM

4.77.2 Purpose

To print parameter values and DMAP messages.

4.77.3 DMAP Calling Sequence

PRTPARM //C,N,α/C,N,β $

4.77.4 Input Data Blocks

None

4.77.5 Output Data Blocks

None

4.77.6 Parameters

α - Integer value (no default value)

β - BCD value (default value = XXXXXXXX)

4.77.7 Method

As a parameter printer, use α = 0. There are two options:

1. β = parameter name will cause the printout of the value of that parameter.

   Example: PRTPARM //C,N,0/C,N,LUSET $

2. β = XXXXXXXX will cause the printout of the values of all parameters in the current XVPS. Since this is the default value, it need not be specified.

   Example: PRTPARM //C,N,0 $

As a DMAP message printer, use $\alpha \neq 0$. There are two options:

1. $\alpha > 0$ causes the printout of the j^{th} message of category β where $j = |\alpha|$ and β is one of the values shown below. (The number of messages available in each category is also shown).

   Example: PRTPARM //C,N,1/C,N,DMAP $

2. $\alpha < 0$ causes the same action as $\alpha > 0$ with the additional action of program termination. Thus, PRTPARM may be used as a fatal message printer.

Example: PRTPARM //C,N,-2/C,N,PLA $

4.77.8 Remarks

1. β is always a value.

2. TABLE OF β CATEGORY VALUES

Rigid Format	Value of Beta	Number of Messages
Static Analysis	STATICS	4
Static Analysis with Inertia Relief	INERTIA	4
Normal Mode Analysis	MØDES	4
Static Analysis with Differential Stiffness	DIFFSTIF	6
Buckling Analysis	BUCKLING	7
Piecewise Linear Analysis	PLA	6
Direct Complex Eigenvalue Analysis	DIRCEAD	5
Direct Frequency and Random Response	DIRFRRD	6
Direct Transient Response	DIRTRD	5
Modal Complex Eigenvalue Analysis	MDLCEAD	6
Modal Frequency and Random Response	MDLFRRD	7
Modal Transient Response	MDLTRD	7
DMAP	DMAP	1

3. For details on error messages for the i^{th} Rigid Format see section 3.(i + 1).2 in the User's Manual.

4.77.9 Subroutines

PRTPRM has no auxiliary subroutines.

4.77.10 Diagnostic Messages

Values of α and β inconsistent with the above under "Method" will result in diagnostic messages from PRTPARM.

4.78 MATRIX MODULE ADD (ADD TWO MATRICES)

4.78.1 Entry Point: DADD

4.78.2 Purpose

To compute [C] = α [A] + β [B].

4.78.3 DMAP Calling Sequence

```
ADD    A,B/C/C,Y,ALPHA=(1.0,2.0)/C,Y,BETA=(3.0,4.0)  $
```

4.78.4 Input Data blocks

A - Any matrix ≠ B

B - Any matrix ≠ A

Note: A and/or B can be purged.

4.78.5 Output Data Blocks

C - Matrix.

The type of C is maximum of the types of A, B, α, β. The shape of C is the shape of A if A is present. Otherwise it is that of B.

Note: C cannot be purged.

4.78.6 Parameters

ALPHA - Input-complex-no default value. This is the scalar multiplier for A.

BETA - Input-complex-no default value. This is the scalar multiplier for B.

Note: If Im(α) or Im(β) = 0.0, the parameter will be considered real.

4.78.7 Method

If [A] is not purged, the number of columns, rows, and form of [C] = number of columns, rows, and form of [A]. Otherwise the [B] descriptors are used. The type of [C] is the maximum compatible type of [A], [B], ALPHA and BETA. ALPHA and BETA are assumed to be real if their imaginary parts are zero.

4.78.8 Subroutines

ADD - See subroutine description, section 3.5.10.

4.78.9 Design Requirements

Open core is defined at /DADDA/.

4.79 MATRIX MODULE MPYAD (MULTIPLY ADD)

4.79.1 Entry Point: DMPYAD

4.79.2 Purpose

MPYAD performs the multiplication of two matrices and, optionally, addition of a third matrix to the product: $[D] = [A][B] \pm [C]$

4.79.3 DMAP Calling Sequence

```
MPYAD     A,B,C/D/C,N,T/C,N,SIGNAB/C,N,SIGNC/C,N,PREC    $
```

4.79.4 Input Data Blocks

A - Left hand matrix in the matrix product [A][B]

B - Right hand matrix in the matrix product [A][B]

C - Matrix to be added to [A][B]

Notes:

1. If no matrix is to added, C must be purged.
2. A, B, C must be physically different data blocks.
3. A and B must not be purged.
4. Either A or B (but not both) may be a NASTRAN diagonal matrix. In this case, C must be purged.

4.79.5 Output Data Block

D - Matrix resulting from the MPYAD operation.

Note: D may not be purged.

4.79.6 Parameters

T - Integer-input-no default. $T = \begin{cases} 1, \text{ perform } [A]^T[B] \\ 0, \text{ perform } [A][B] \end{cases}$

SIGNAB - Integer-input-no default. SIGNAB = {+1, perform [A][B]; -1, perform -[A][B]}

SIGNC - Integer-input-no default. SIGNC = {+1, add [C]; -1, subract [C]}

PREC - Integer-input-no default. PREC = {1, elements of [D] will be output in single-precision. 2, elements of [D] will be output in double-precision.}

4.79.7 Examples

1. [D] = [A][B] + [C] (D double precision)

```
MPYAD     A,B,C/D/C,N,0/C,N,1/C,N,2  $
```

2. $[D] = [A]^T[B] - [C]$ (D single precision)

```
MPYAD     A,B,C/D/C,N,1/C,N,1/C,N,-1/C,N,1  $
```

3. [D] = -[A][B] (D double precision)

```
MPYAD     A,B,/D/C,N,0/C,N,-1/C,N,0/C,N,2  $
```

4.79.8 Method

DMPYAD reads the trailers for the data blocks A, B and C. /MPYADX/ is initialized. If neither [A] nor [B] is diagonal, MPYAD is called, the trailer for D is written, and the module exits. Otherwise, /DMPYX/ is initialized, and DMPY is called to perform the diagonal multiplication. If the matrix [C] is present, /ADDX/ is initialized, and ADD is called to perform the matrix addition. The trailer for D is written and the module exits.

4.79.9 Subroutines

DMPYAD calls the following matrix operations:

MPYAD (see section 3.5.12 for details)

DMPY (see section 3.5.21 for details)

ADD (see section 3.6.10 for details)

4.79.10 Design Requirements

4.79.10.1 Allocation of core storage

See descriptions for MPYAD, DMPY and ADD.

4.79.10.2 Environment

The module MPYAD consists of one subroutine, DMPYAD. One scratch file is used. Three common blocks define open core, one for each of the three overlay segments containing the matrix operations:

/MPYA1D/ included at end of segment containing MPYAD.

/MPYA2D/ included at end of segment containing DMPY.

/MPYA3D/ included at end of segment containing ADD.

4.80 MATRIX MODULE SØLVE (SOLVES THE MATRIX EQUATION [A][X] = [B])

4.80.1 Entry Point: SØLVE

4.80.2 Purpose

To solve the matrix equation,

$$[A][X] = \pm [B] \tag{1}$$

4.80.3 DMAP Calling Sequence

```
SØLVE    A,B/X/V,Y,SYM/V,Y,SIGN/V,Y,PREC/V,Y,TYPE    $
```

4.80.4 Input Data Blocks

A - A square real or complex matrix.

B - A rectangular matrix.

Note: If B is purged, the identity matrix is assumed.

4.80.5 Output Data Blocks

X - A rectangular matrix.

4.80.6 Parameters

SYM - Input-integer-default = 0

- 0 - use unsymmetric decomposition
- 1 - use symmetric decomposition.
- -1 - try symmetric decomposition, use unsymmetric if [A] is singular.

SIGN - input-integer-default = 1

- 1 - solve [A][X] = [B]
- -1 - solve [A][X] = -[B]

PREC - Input-integer-default = 1

- 1 - use single precision arithmetic
- 2 - use double precision arithmetic

TYPE - Input-integer-default = 1

- 1 - output type of matrix [X] is real single precision
- 2 - output type of matrix [X] is real double precision

3 - output type of matrix [X] is complex single precision

4 - output type of matrix [X] is complex double precision

4.80.7 Method

Depending upon the SYM flag and the type of [A], either SDCØMP CDCØMP, or DECØMP is called to form

$$[A] = [L][U] \ . \tag{2}$$

FBS or GFBS is called to solve

$$[L][Y] = \pm [B] \ , \tag{3}$$

and

$$[U][X] = [Y] \ . \tag{4}$$

4.80.8 Subroutines

The above mentioned subroutines are the only ones called by SØLVE and are documented in section 3.5.

4.80.9 Design Requirements

The appropriate subroutines should be referenced for the design requirements peculiar to each routine.

4.80.10 Diagnostic Messages

The individual routines should be referred to for diagnostic messages.

4.81 MATRIX MODULE DECØMP (MATRIX DECOMPOSITION)

4.81.1 Entry Point: DDCØMP

4.81.2 Purpose

To decompose a square matrix [A] into lower [L] and upper [U] triangular factors.

4.81.3 DMAP Calling Sequence

```
DECØMP    A/L,U/V,Y,KSYM/V,Y,CHØLSKY/V,N,MINDIAG/V,N,DET/V,N,PØWER/V,N,SING    $
```

4.81.4 Input Data Blocks

A - A square matrix.

4.81.5 Output Data Blocks

L - Lower triangular factor of [A].

U - Non-standard upper triangular factor of [A].

4.81.6 Parameters

KSYM - Input-integer-no default. 1, use symmetric decomposition. 0, use unsymmetric decomposition.

CHØLSKY - Input-integer-default = 0. 1, use Cholesky decomposition. 0, do not use Cholesky decomposition.

MINDIAG - Output-real-no default. The minimum diagonal term of [U].

DET - Output-complex single precision-no default. The scaled value of the determinant of [A].

PØWER - Output-integer-no default. Integer PØWER of 10 by which DET should be multiplied to obtain the determinant of [A].

SING - Output-integer-no default. SING is set to -1 if [A] is singular.

4.81.7 Method

Depending upon the type of [A] and the KSYM flag, a calling sequence is set up, and either

CDCØMP, DECØMP, or SDCØMP is called.

4.81.8 Subroutines

The major subroutines used are DECØMP, CDCØMP and SDCØMP. Descriptions of these subroutines can be found in sections 3.5.15, 3.5.16, and 3.5.14 respectively.

4.81.9 Design Requirements

The individual subroutine writeups should be consulted for the particular restrictions of each routine.

4.81.10 Diagnostic Messages

See the appropriate subroutine descriptions.

4.82 MATRIX MODULE FBS (FORWARD-BACKWARD SUBSTITUTION)

4.82.1 Entry Point: DFBS

4.82.2 Purpose

To solve the equation,

$$[L]\,[U]\,[X] = \pm\,[B] \tag{1}$$

where [L] and [U] are the upper and lower triangular factors obtained via matrix module DECØMP.

4.82.3 DMAP Calling Sequence

```
FBS    L,U,B/X/C,N,A/C,N,B/C,N,C/C,N,D    $
```

4.82.4 Input Data Blocks

L,U - Matrices output from module DECØMP.

B - Rectangular matrix.

4.82.5 Output Data Blocks

X - Rectangular matrix.

4.82.6 Parameters

A - Input-integer-no default.
- 1 matrix [L][U] is symmetric
- 0 matrix [L][U] is unsymmetric

B - Input-integer-no default.
- 1 solve [L][U][X] = [B]
- -1 solve [L][U][X] = -[B]

C - Input-integer-no default.
- 1 - use single precision arithmetic
- 2 - use double precision arithmetic

D - Input-integer-no default.
- 1 - output [X] in single precision
- 2 - output [X] in double precision

4.82.7 Method

Depending upon the value of the parameter A, either FBS or GFBS is called.

4.82.8 Subroutines

The above routines are the only ones called by DFBS. Their descriptions are given in section 3.5.17 for FBS and 3.5.19 for GFBS.

4.82.9 Design Requirements

The appropriate routines should be referenced for their individual requirements.

4.82.10 Diagnostic Messages

The individual subroutines should be referred to for the messages.

4.83 MATRIX MODULE PARTN (PARTITION A MATRIX)

4.83.1 Entry Point: PARTNX.

4.83.2 Purpose

To partition [A] into [A11], [A12], [A21] and [A22]:

$$[A] \Longrightarrow \left[\begin{array}{c|c} A11 & A21 \\ \hline A12 & A22 \end{array}\right]. \qquad (1)$$

4.83.3 DMAP Calling Sequence

```
PARTN    A,RP,CP/A11,A12,A21,A22/V,Y,SYM/V,Y,TYPE/V,Y,FORM1/V,Y,FORM2/V,Y,FORM3/V,Y,FORM4 $
```

4.83.4 Input Data Blocks

A - Matrix to be partitioned.

RP - Row partitioning vector - single precision column vector.

CP - Column partitioning vector - single precision column vector.

Notes:

1. If A is purged, PARTN returns.

2. If RP is purged, A is partitioned as follows:

$$[A] \Longrightarrow \left[\begin{array}{c} A11 \\ \hline A12 \end{array}\right]. \qquad (2)$$

3. If CP is purged and SYM > 0, A is partitioned as follows:

$$[A] \Rightarrow [A11 \mid A21]. \qquad (3)$$

4. If CP is purged and SYM ≤ 0, A is partitioned as follows:

$$[A] \Rightarrow \left[\begin{array}{c|c} A11 & A21 \\ \hline A12 & A22 \end{array}\right], \qquad (4)$$

where RP is used as both the row and column partitioner.

5. RP and CP cannot both be purged.

4.83.5 Output Data Blocks

A11 - Partition of A.

A12 - Partition of A.

A21 - Partition of A.

A22 - Partition of A.

Notes:

1. Any or all output data blocks can be purged.
2. For the shape of outputs (number of rows and columns) see section 4.83.7 below.

4.83.6 Parameters

SYM - Input-integer-no default. SYM chooses between a symmetric partition and an unsymmetric partition. If SYM ≤ 0, CP is used as RP. If SYM > 0, CP and RP are distinct.

TYPE - Input-integer-no default. TYPE of output matrices. 0 < TYPE ≤ 8.

FØRM1 - Input-integer-no default. Form of A11.

FØRM2 - Input-integer-no default. Form of A12.

FØRM3 - Input-integer-no default. Form of A21.

FØRM4 - Input-integer-no default. Form of A22.

4.83.7 Method

Let N1 = number of non-zero terms in RP.

Let N2 = number of non-zero terms in CP.

Let NRØWA = number of rows in A.

Let NCØLA = number of columns in A.

CASE: 1 RP purged.

A11 is a (NRØWA-N2) x NCØLA matrix.

A12 is a N2 x NCØLA matrix.

A21 is not written.

A22 is not written.

CASE 2: CP purged and SYM > 0.

A11 is a NRØWA x (NCØLA - N1) matrix.

A12 is not written.

A21 is a NRØWA x N1 matrix.

A22 is not written.

CASE 3: CP purged and SYM $\leq$ 0.

A11 is a (NRØWA - N1) x (NCØLA - N1) matrix.

A12 is a N1 x (NCØLA - N1) matrix.

A21 is a (NRØWA - N1) x N1 matrix.

A22 is a N1 x N1 matrix.

CASE 4: Neither RP nor CP purged.

A11 is a (NRØWA - N2) x (NCØLA - N1) matrix.

A12 is a N2 x (NCØLA - N1) matrix.

A21 is a (NRØWA - N2) x N1 matrix.

A22 is a N2 x N1 matrix.

In general if $a_{ij} \in [A]$, then:

$a_{ij} \in [A11]$ if RP(J) = CP(I) = 0

$a_{ij} \in [A12]$ if CP(I) $\neq$ 0, RP(J) = 0

$a_{ij} \in [A21]$ if CP(I) = 0, RP(J) $\neq$ 0

$a_{ij} \in [A22]$ if CP(I) $\neq$ 0, RP(J) $\neq$ 0

4.83.8 Subroutines

PARTN - See subroutine description, section 3.5.6.

4.83.9 Design Requirements

Open core is defined at /PARTN1/.

4.84 MATRIX MODULE MERGE (MERGE MATRICES TOGETHER)

4.84.1 Entry Point: MERGEX

4.84.2 Purpose

To form

$$[A] \Leftarrow \left[\begin{array}{c|c} A11 & A21 \\ \hline A12 & A22 \end{array}\right]. \qquad (1)$$

4.84.3 DMAP Calling Sequence

```
MERGE    A11,A12,A21,A22,RP,CP/A/V,Y,SYM/V,Y,TYPE/V,Y,FØRM  $
```

4.84.4 Input Data Blocks

A11 - Matrix ≠ A12, A21, A22.

A12 - Matrix ≠ A11, A21, A22.

A21 - Matrix ≠ A11, A12, A22.

A22 - Matrix ≠ A11, A12, A21.

RP - Row partitioning vector - single precision vector.

CP - Column partitioning vector - single precision vector.

Notes:

1. Any or all of A11, A12, A21, A22 can be purged which implies [AIJ] = [0].
2. RP and CP cannot both be purged.
3. See method section for meaning when RP or CP is purged.

4.84.5 Output Data Blocks

A - Merged matrix from A11, A12, A21, A22.

Notes: A cannot be purged.

4.84.6 Parameters

SYM - Input-integer-no default. SYM $\leq$ 0, CP is used as RP. SYM > 0, CP and RP are distinct.

TYPE - Input-integer-no default. Type of A. 1 implies A is real single precision, 2 implies A is real double precision, 3 implies A is complex single precision, 4 implies A is complex double precision.

FØRM - Input-integer-no default. Form of A (see section 2.2).

4.84.7 Method

MERGE is the inverse of PARTN in the sense that if A11, A12, A21, A22 were produced by PARTN using RP, CP, FØRM, SYM, and TYPE from A, MERGE will reproduce A. See PARTN (section 4.83) for options on RP, CP and SYM.

4.84.8 Subroutines

MERGE - See subroutine description, section 3.5.6.

4.84.9 Design Requirements

Open core is defined at /MERGE1/.

4.85 MATRIX MODULE TRNSP (TRANSPOSE A MATRIX).

4.85.1 Entry Point: DTRANP

4.85.2 Purpose

To form $[A]^T$ given [A].

4.85.3 DMAP Calling Sequence

```
TRNSP    A/AT  $
```

4.85.4 Input Data Blocks

A - Any matrix data block.

Note: If [A] is purged, TRNSP returns.

4.85.5 Output Data Blocks

AT - The matrix transpose of [A].

Note: AT cannot be purged.

4.85.6 Parameters

None.

4.85.7 Method

Subroutine TRNSP is called.

4.85.8 Subroutines

TRNSP - See subroutine description, section 3.5.25.

4.85.9 Design Requirements

Open core is defined at /DTRANX/. Eight scratch files are used.

4.86 MATRIX MODULE SMPYAD (STRING MULTIPLY ADD)

4.86.1 Entry Point: SMPYAD

4.86.2 Purpose

To multiply a series of matrices together.

4.86.3 DMAP Calling Sequence

```
SMPYAD    A,B,C,D,E,F/G/C,N,N/C,N,SIGNX/C,N,SIGNF/C,N,PG/C,N,TA/C,N,TB/C,N,TC/C,N,TD $
```

4.86.4 Input Data Blocks

A, B, C, D, E - Up to 5 matrices to be multiplied together, from left to right.

F - Matrix to be added to the above product.

Notes:

1. If one of the five multiplication matrices is required in the product (see parameter "N" below) and is purged, the multiplication will not be done.

2. If the F matrix is purged, no matrix will be added to the product.

4.86.5 Output Data Blocks

G - Resultant matrix (may not be pre-purged).

4.86.6 Parameters

N - Number of matrices involved in the product - integer - input.

SIGNX - Sign of the product matrix (e.g., [A][B][C][D][E]) - integer - input. 1 for plus, -1 for minus.

SIGNF - Sign of the matrix tc be added to the product matrix - integer - input. 1 for plus, -1 for minus.

PG - Output precision of the final result - integer - input. 1 for single precision, 2 for double precision.

TA, TB, TC, TD - Transpose indicators for the [A][B][C] and [D] matrices (1 if transposed matrix to be used in the product; 0 if untransposed) - integer - input.

Note: All the parameters except "N" have default values. They are these:

1. SIGNX = 1 (sign of product is plus)
2. SIGNF = 1 (sign of added matrix is plus)
3. PG = 1 (single precision result)
4. TA, TB, TC, TD = 0 (use untransposed [A], [B], [C], and [D] matrices in the product)

4.86.7 Method

The method is the same as for the MPYAD module with one exception and one addition:

1. None of the matrices may be diagonal.
2. Except for the final product, all intermediate matrix products are generated in double precision.

The matrices are multiplied together from right-to-left, i.e., the first product calculated is the product of matrix n-1 and matrix n.

4.86.8 Subroutines

MPYAD is called (see section 3.5.12 for details).

4.86.9 Design Requirements

1. Two scratch files are required.
2. Open core is the /MPYADX/ common block, the same one as used by the MPYAD module, (see section 4.79).

4.87 STRUCTURAL ELEMENT DESCRIPTIONS

The finite structural element subroutines used in NASTRAN have a number of different calculations associated with them. These subroutines are found in the modules SMA1, SMA2, SSG1, SDR2, DSMG1, PLA3 and PLA4.

The element subroutines in the SMA1 (Structural Matrix Assembler - Phase 1) module generate element stiffness matrix partitions. The stiffness matrix, [K], for a structural element consists of a 6 by 6 partition for each combination of the connected grid points. For example, a RØD element is connected to two grid points, "a" and "b". The stiffness matrix partitions are: $[K_{aa}]$, $[K_{ab}]$, $[K_{ba}]$ and $[K_{bb}]$. A triangular element (e.g., TRMEM) is connected to three points. It will generate nine partitions: $[K_{aa}]$, $[K_{ab}]$, $[K_{ac}]$, $[K_{ba}]$, $[K_{bb}]$, $[K_{bc}]$, $[K_{ca}]$, $[K_{cb}]$ and $[K_{cc}]$. In order to generate a particular partition, $[K_{ij}]$, it is often necessary to generate [K]. However, only those partitions $[K_{ij}]$, where i is the pivot point (see section 1.8) and j = 1,2,...,n (n being the number of grid points associated with the element), are output by an element stiffness matrix generation subroutine, e.g., KRØD. These partitions are output from an element subroutine in the form of calls to the "insertion" subroutine SMA1B (see section 4.27). There is one call for each 6 by 6 partition if the element is a structural element, and one call for each 1 by 1 "partition" if the element is a scalar element. The unused partitions are recalculated and used when $j \neq i$ appears as a pivot point in a subsequent ECPT record. An alternate procedure for matrix generation, which is not used, would be to calculate all of the element matrices once and store them on an auxiliary storage unit for use when needed. The alternate procedure is less efficient for large problems, where efficiency really counts, because the recalculation time is less than the time required to recover element matrices from the auxiliary unit.

Element structural damping matrices, $[K^4]$, are proportional to the element stiffness matrices, the proportionality constant being g_e, the structural damping coefficient input on a material (e.g., MAT1) bulk data card. An element stiffness matrix generation routine, e.g., KRØD, of module SMA1 will output, through the calling sequence to subroutine SMA1B: 1) an element stiffness matrix partition, 2) the structural damping coefficient, and 3) a flag, which will signal SMA1B that the scalar multiplication of the matrix by the structural damping coefficient is to take place.

The element subroutines (e.g., MRØD, MCØNMX) in the SMA2 (Structural Matrix Assembler - Phase 2) module generate element mass matrix partitions. The remarks in the second paragraph above

concerning element stiffness matrix partitions apply here also when the reader makes the substitutions: "mass" for "stiffness", "[M]" for "[K]", "MRØD" for "KRØD", and "SMA2B" for "SMA1B".

Only the elements VISC and DAMPi generate viscous damping terms which contribute to the damping matrix, $[B_{gg}]$, and conversely, the only elements which contribute to $[B_{gg}]$ are the VISC and DAMPi elements. These terms are calculated in module SMA2. The damping matrix partitions are passed to subroutine SMA2B in a fashion similar to that for mass matrix partitions.

Element static loading functions due to temperature and enforced deformations are generated in the SSG1 (Static Solution Generator - Phase 1) module, and the mathematical descriptions for these functions are given in this section (4.87). (See the Module Functional Description for SSG1, section 4.41, for the equations governing direct applied loads and gravity loads.) The output of an element routine are load vectors which are placed in the $\{P_g\}$ load vector (see section 4.41).

Element stresses and forces due to displacements are calculated in the SDR2 (Stress Data Recovery - Phase 2) module. These calculations are performed in two phases. Phase 1 generates element stress matrices for each element for which the user has requested element stress and/or force output. These element stress matrices are written on a scratch file for use in phase 2. In phase 2, the displacement vector for the current subcase is read into core, and, for each element for which stress and/or force output is requested, the corresponding element stress matrix is read and passed to the phase 2 element subroutine. The phase 2 element subroutine then calculates element stresses and forces. A list of the stresses and forces output in phase 2 for each element is given in sections 2.3.51 and 2.3.52 respectively.

Differential stiffness matrix partitions are calculated for some elements. These are calculated in module DSMG1 (Differential Stiffness Matrix Generator - Phase 1) for large displacement analysis and buckling problems. The output of an element routine of the DSMG1 module are the 6 by 6 differential stiffness matrix partitions, $[K^d_{ij}]$, where i is the pivot point. The "insertion" subroutine for module DSMG1, similar to subroutine SMA1B of module SMA1, is DS1B.

Nonlinear, plastic effects in the structure may be determined by solving for the element stress and modifying the elastic properties of an element in an iterative loop. Element stresses are calculated in the PLA3 (Piecewise Linear Analysis - Phase 3) module, and element stiffness matrices with modified elastic properties are calculated in the PLA4 (Piecewise Linear Analysis - Phase 4). The outputs of an element subroutine of the PLA3 module are: 1) element stresses,

which have the same formats as the element stresses output from a phase 2 element subroutine of module SDR2, and 2) updated incremental stress data in the ESTNL1 data block, which are used as input to the PLA3 module in the next pass of the Piecewise Linear Analysis (PLA) Rigid Format DMAP loop. The outputs of an element subroutine of the PLA4 module are: 1) element stiffness matrix partitions (the remarks on element stiffness matrix partitions in the second paragraph apply here as well, except that the "insertion" subroutine is PLA4B) and 2) updated incremental stress data in the ECPTNL1 data block, which are used as input to the PLA4 module in the next pass of the PLA Rigid Format DMAP loop.

The following data are needed to generate the element matrices in the above modules.

1. Element Connection and Properties Table (ECPT) Data.

2. Transformation matrices, $[T_i]$, from the global coordinate system to the basic coordinate system.

3. Material Property Data.

4. Element Deformation Data (used only in modules SSG1, SDR2 and DSGM1).

5. Grid Point Temperature Data (used only in modules SSG1, SDR2 and DSGM1).

The ECPT data are input to an element subroutine by a module driver from the ECPT data block or the EST (Element Summary Table) data block. The data in each of these data blocks are identical, from an individual element subroutine point of view. The ECPT data block is used in modules SMA1, SMA2 and DSGM1; the EST data block is used in modules SSG1 and SDR2. For the special case of Piecewise Linear Analysis, the ECPTNL (Element Connection and Properties Table for Nonlinear Elements) data block is used in module PLA4, and the ESTNL (Element Summary Table for Nonlinear Elements) data block is used in module PLA3. The ECPT and EST data blocks are generated in the Table Assembler (TA1) module (see section 4.26) from the following data blocks: ECT (Element Connection Table, section 2.3.4.1), EPT (Element Property Table, section 2.3.2.5), BGPDT (Basic Grid Point Definition Table, section 2.3.3.5), and GPTT (Grid Point Temperature Table, section 2.3.7.2).

The ECPT data for an element consist of four separate parts: 1) connection data 2) property data, 3) basic grid point definition data and 4) the element temperature for material properties. The connection data consists of data on a connection bulk data card (e.g., CRØD), except for the property identification number (the property identification number on the connection and property

cards is used only to relate the two cards during the assembly of the ECPT and EST data blocks, and it does not appear in either the connection data or property data). Note also that grid point identification numbers have been converted to internal numbers, Scalar Index List (SIL) numbers, which correspond to degree of freedom numbers. Property data consist of data on a property bulk data card (e.g., PRØD) with the above noted exception. Basic grid point definition data consist of, for each grid point connecting the element, 1) the identification number of the coordinate system in which displacements are defined at the grid point and 2) the coordinates of the grid point in the basic coordinate system. The element temperature for material properties is the average of the temperatures at all grid points defining the element. The temperature for material properties, t_μ, for a RØD element, for example, is 1/2 $(t_a + t_b)$, where t_a and t_b are the temperatures at the grid points a and b which define the end points of the RØD; similarly, for a triangular element, say TRMEM, the temperature for material properties would be 1/3 $(t_a + t_b + t_c)$, etc. This temperature is computed in the Table Assembler module from the grid point temperatures in the GPTT data block and the set identification number, n, from the TEMPERATURE(MATERIAL) = n card in the user's Case Control Deck. Note that n is transmitted to the Table Assembler via the tenth word of /SYSTEM/ (see section 2.4.1.8).

The transformation matrices, $[T_i]$, from the global coordinate system to the basic coordinate system, are supplied to an element subroutine by the utility routines TRANSD and TRANSS. These utility routines use the CSTM (Coordinate System Transformation Matrices, section 2.3.3.4) data block in conjunction with the basic grid point definition data at a point i to compute $[T_i]$. Hence all modules which deal with element calculations require the CSTM data block as input. TRANSD returns, to an element subroutine, a double precision matrix, $[T_i]$, used by element routines in the following modules which use double precision arithmetic: SMA1, SMA2, DSGM1 and PLA4; TRANSS returns, to an element subroutine, a single precision matrix, $[T_i]$, used by element routines in the following modules which use single precision arithmetic: SSG1, SDR2 and PLA3.

Material property data are contained in the MPT (Material Properties Table, section 2.3.2.6) and the DIT (Direct Input Tables, section 2.3.2.7) data blocks. Both of these data blocks are output from the IFP (Input File Processor) Preface module. The utility routine MAT (see section 3.4.36) fetches required material property data for element routines. These data are returned in single precision form.

Element deformation data are contained in the EDT (Element Deformation Table, section 2.3.2.8)

data block which is output by the IFP Preface module. Element deformation data is admissible only for the RØD (including CØNRØD), TUBE and BAR elements.

Grid point temperature data are contained in the GPTT (Grid Point Temperature Table, section 2.3.7.2), which is output by the GP3 (Geometry Processor - Phase 3) module. The temperature data contained in this data block are used for static loading functions due to temperature.

Table 1 on the following page gives reference to the Theoretical and User's Manuals where more information on the elements can be found.

Table 1. Structural Element References.

Bulk Data Connection Card Mnemonic	Programmer's Manual Reference	User's Manual Reference	Theoretical Manual Reference
CBAR	4.87.2	1.3.2	5.2, 7.2
CCØNEAX	4.87.9	1.3.6	5.9
CDAMP1	4.87.7	1.3.8	5.6
CDAMP2	4.87.7	1.3.8	5.6
CDAMP3	4.87.7	1.3.8	5.6
CDAMP4	4.87.7	1.3.8	5.6
CELAS1	4.87.7	1.3.8	5.6
CELAS2	4.87.7	1.3.8	5.6
CELAS3	4.87.7	1.3.8	5.6
CELAS4	4.87.7	1.3.8	5.6
CMASS1	4.87.7	1.3.8	5.6
CMASS2	4.87.7	1.3.8	5.6
CMASS3	4.87.7	1.3.8	5.6
CMASS4	4.87.7	1.3.8	5.6
CØNM1	4.87.8	1.2.3	5.5
CØNM2	4.87.8	1.2.3	5.5
CØNRØD	4.87.1	1.3.3	5.2, 7.2
CQDMEM	4.87.4	1.3.5	5.8, 7.3
CQDPLT	4.87.5	1.3.5	5.8
CQUAD1	4.87.6	1.3.5	5.8, 7.3
CQUAD2	4.87.6	1.3.5	5.8, 7.3
CRØD	4.87.1	1.3.3	5.2, 7.2
CSHEAR	4.87.3	1.3.4	5.3
CTØRDRG	4.87.12	1.3.7	5.11
CTRAPRG	4.87.11	1.3.7	5.10
CTRBSC	4.87.5	1.3.5	5.8
CTRIA1	4.87.6	1.3.5	5.8, 7.3
CTRIA2	4.87.6	1.3.5	5.8, 7.3
CTRIARG	4.87.10	1.3.7	5.8
CTRMEM	4.87.4	1.3.5	5.8, 7.3
CTRPLT	4.87.5	1.3.5	5.8
CTUBE	4.87.1	1.3.3	5.2, 7.2
CTWIST	4.87.3	1.3.4	5.3
CVISC	4.87.14	1.3.3	5.2

Note The bulk data connection and property card descriptions in section 2 of the User's Manual should also be consulted.

4.87.1 The RØD, CØNRØD and TUBE Elements

4.87.1.1 Input Data for the RØD, TUBE, CØNRØD Elements

1. The ECPT/EST entries for the RØD and CØNRØD are:

Symbol	Description
SIL_a, SIL_b	Scalar indices for grid points a and b
N_a, X_a, Y_a, Z_a N_b, X_b, Y_b, Z_b	Local coordinate system number and basic coordinates of grid points
Mat I. D.	Material identification number
A	Cross-section area
J	Polar inertia
μ	Nonstructural mass per unit length
C	Shear stress coefficient
t_μ	Temperature for material properties

2. The TUBE element has the same characteristics as the RØD except for different input properties. The TUBE has d, the outside diameter, and t, the thickness, given. The conversion to RØD properties is:

$$A = \pi(d - t)t$$

$$J = \frac{1}{4}A\left((d - t)^2 + t^2\right)$$

$$C = \frac{d}{2}$$

3. Coordinate system data

Given N_a, X_a, Y_a, Z_a, N_b, X_b, Y_b and Z_b and the CSTM (Coordinate System Transformation Matrices) data block, the 3 by 3 global-to-basic coordinate transformation matrices $[T_a]$ and $[T_b]$ are calculated using the utility routine TRANSD or TRANSS.

4. Material data

Given the "MAT I.D." and t_μ, the material routine, MAT, returns the following data:

- E - Modulus of Elasticity
- G - Shear Modulus
- ν - Poisson's ratio
- ρ - Density
- α - Thermal expansion coefficient
- T_o - Reference temperature
- g_e - Structural damping ratio
- σ_t - Stress limit, tension
- σ_c - Stress limit, compression
- σ_s - Stress limit, shear

4.87.1.2 Stiffness Matrix Calculation (Subroutines KRØD and KTUBE of Module SMA1)

1. Calculate the length of member, (ℓ):

$$\ell = \sqrt{(X_a - X_b)^2 + (Y_a - Y_b)^2 + (Z_a - Z_b)^2} \quad . \tag{1}$$

2. Calculate a normalized direction vector {n} in basic coordinates:

$$\begin{Bmatrix} n_1 \\ n_2 \\ n_3 \end{Bmatrix} = \frac{1}{\ell} \begin{Bmatrix} X_a - X_b \\ Y_a - Y_b \\ Z_a - Z_b \end{Bmatrix} \quad . \tag{2}$$

3. Form the extensional stiffness matrix, $[D_\ell]$:

$$[D_\ell] = \frac{AE}{\ell} \begin{bmatrix} n_1^2 & n_1 n_2 & n_1 n_3 \\ n_1 n_2 & n_2^2 & n_2 n_3 \\ n_1 n_3 & n_2 n_3 & n_3^2 \end{bmatrix} \quad . \tag{3}$$

4. Form the torsional stiffness matrix $[D_r]$:

$$[D_r] = \frac{GJ}{\ell}\begin{bmatrix} n_1^2 & n_1n_2 & n_1n_3 \\ n_1n_2 & n_2^2 & n_2n_3 \\ n_1n_3 & n_2n_3 & n_3^2 \end{bmatrix} . \tag{4}$$

5. $[T_a]$ and $[T_b]$ are the matrices which transform displacement components in the global coordinate system to the basic system.

6. Transforming to global coordinates and combining the results give the partitions of the element stiffness matrix:

$$[k_{aa}] = \left[\begin{array}{c|c} T_a^T D_\ell T_a & 0 \\ \hline 0 & T_a^T D_r T_a \end{array}\right] , \tag{5}$$

$$[k_{ab}] = -\left[\begin{array}{c|c} T_a^T D_\ell T_b & 0 \\ \hline 0 & T_a^T D_r T_b \end{array}\right] , \tag{6}$$

$$[k_{bb}] = \left[\begin{array}{c|c} T_b^T D_\ell T_b & 0 \\ \hline 0 & T_b^T D_r T_b \end{array}\right] , \tag{7}$$

$$[k_{ba}] = [k_{ab}]^T . \tag{8}$$

The element damping matrices are equal to the stiffness matrices times g_e, the structural damping coefficient.

4.87.1.3 Lumped Mass Matrix Calculation (Subroutines MRØD and MTUBE of Module SMA2)

The total mass of the element, m, is

$$m = \rho A\ell + \mu\ell . \tag{9}$$

The partitions of the element mass matrix are:

$$[M_{aa}] = [M_{bb}] = \frac{m}{2} \begin{bmatrix} 1 & & & 0 & \\ & 1 & & & \\ & & 1 & & \\ \hline & & & 0 & \\ 0 & & & 0 & \\ & & & & 0 \end{bmatrix}, \tag{10}$$

$$[M_{ab}] = [M_{ba}] = [0] . \tag{11}$$

4.87.1.4 Element Load Calculations (Subroutine EDTL of Module SSG1)

The element loading calculations are calculated using the EST data, the grid point loading temperatures t_a and t_b, and the enforced deformation δ.

1. Calculate ℓ, $\{n\}$, $[T_a]$ and $[T_b]$ as in section 4.87.1.2.

2. Calculate:

$$\bar{t} = (t_a + t_b)/2 - T_o, \tag{12}$$

$$\{P_a\} = \frac{EA}{\ell} \begin{bmatrix} [T_a]^T \\ \hline 0 \end{bmatrix} \{n\} (\delta + \alpha \ell \bar{t}), \tag{13}$$

$$\{P_b\} = -\frac{EA}{\ell} \begin{bmatrix} [T_b]^T \\ \hline 0 \end{bmatrix} \{n\} (\delta + \alpha \ell \bar{t}). \tag{14}$$

3. $\{P_a\}$ and $\{P_b\}$ are placed in the load vector in positions corresponding to points a and b.

4.87.1.5 Element Stress Calculations (Subroutines SRØD1 and SRØD2 of Module SDR2).

The stress functions calculated in phase 1 (Subroutine SRØD1) are:

$$[S_a^t] = \frac{E}{\ell} \{n\}^T [T_a], \qquad (1x3); \tag{15}$$

$$[S_b^t] = -\frac{E}{\ell}\{n\}^T[T_b], \quad (1x3); \tag{16}$$

$$[S_a^r] = \frac{GC}{\ell}\{n\}^T[T_b], \quad (1x3); \tag{17}$$

$$[S_b^r] = -\frac{GC}{\ell}\{n\}^T[T_b], \quad (1x3). \tag{18}$$

$$S_T = -\alpha E, \tag{19}$$

$$S_\delta = -\frac{E}{\ell}. \tag{20}$$

The miscellaneous constants A, $\frac{J}{C}$, T_o, σ_t, σ_c and σ_s are also saved for phase 2 calculations.

Note J/C is set to zero if C = 0. The superscripts t and r denotes translational and rotational stress matrices respectively.

The stress and force values are calculated in phase 2 (Subroutine SRØD2) using the displacement vectors $\{u_a\}$ and $\{u_b\}$, the loading temperatures t_a and t_b, and the enforced deformation δ. Note that $\{u_i^t\}$ and $\{u_i^r\}$ (i = a, b) denote the 3 by 1 translational and rotational components of $\{u_i\}$.

1. Partition

$$\{u_a\} \Longrightarrow \left\{\begin{array}{c} u_a^t \\ \hline u_a^r \end{array}\right\}, \tag{21}$$

$$\{u_b\} \Longrightarrow \left\{\begin{array}{c} u_b^t \\ \hline u_b^r \end{array}\right\}. \tag{22}$$

2. The stresses are:

$$\sigma = [S_a^t]\{u_a^t\} + [S_b^t]\{u_b^t\} + S_\delta\delta + S_T\left[\frac{1}{2}(t_a + t_b) - T_o\right], \tag{23}$$

$$\tau = [S_a^r]\{u_a^r\} + [S_a^r]\{u_b^r\}. \tag{24}$$

The margins of safety (M.S.) are calculated as follows:

If $\sigma \geq 0$ then:

$$\text{M.S.} = \begin{cases} \dfrac{\sigma_t}{\sigma} - 1 \ , \ \sigma_t > 0 \\ \text{Integer "1"}, \ \sigma_t \leq 0 \end{cases} , \tag{25}$$

If $\sigma < 0$, then:

$$\text{M.S.} = \begin{cases} \dfrac{\sigma_c}{\sigma} + 1 \ , \ \sigma_c > 0 \\ \text{Integer "1"}, \ \sigma_c \leq 0 \end{cases} . \tag{26}$$

The forces are:

$$P = A\sigma \ , \tag{27}$$

$$T = \frac{J}{C}\tau \ . \tag{28}$$

4.87.1.6 Differential Stiffness Matrix Calculation (Subroutine DRØD of Module DSMG1)

The data input from the ECPT, MPT and CSTM data blocks are as listed in section 4.87.1.1. The following variables are calculated in the same manner as those for the stiffness matrix variables (section 4.87.1.2):

ℓ	length of rod
n_1, n_2, n_3	The direction cosines of the rod axis (+ from b to a) in the basic coordinate system
$[T_a]$, $[T_b]$	The transformation matrices from global coordinates to the basic coordinate system at the grid points.

Only the linear translational displacements at the grid points are extracted from the displacement vector. Call these $\{u_a^t\}$ and $\{u_b^t\}$ (3x1 single precision vectors.)

1. Calculate the axial load in the element (+ implies tension):

$$\frac{F_x}{\ell} = \frac{AE}{\ell^2}\left\{\{n\}^T[[T_a]\{u_a^t\} - [T_b]\{u_b^t\}] - \delta - \alpha\ell\left(\frac{t_a + t_b}{2}\right) - T_o\right\} . \tag{29}$$

2. A pair of axes perpendicular to the rod axis is constructed. Select the smallest component of {n}. Define n_i as the component of {n} which is the smallest; let j and k be the other two components. Construct {m} such that

$$m_i = 1, \quad m_j = m_k = 0. \tag{30}$$

Let

$$\{y\} = \frac{\{m\} \times \{n\}}{|\{m\} \times \{n\}|} , \tag{31}$$

and

$$\{z\} = \frac{\{n\} \times \{y\}}{|\{n\} \times \{y\}|} , \tag{32}$$

where x denotes the cross product.

The actual partitions of the differential stiffness matrix relating to displacements in global coordinates are:

$$[k_{aa}^d] = \frac{F_x}{\ell}[T_a]^T[\{y\}\{y\}^T + \{z\}\{z\}^T][T_a] , \tag{33}$$

$$[k_{ab}^d] = \frac{-F_x}{\ell}[T_a]^T[\{y\}\{y\}^T + \{z\}\{z\}^T][T_b] , \tag{34}$$

$$[k_{bb}^d] = \frac{F_x}{\ell}[T_b]^T[\{y\}\{y\}^T + \{z\}\{z\}^T][T_b] , \tag{35}$$

$$[k_{ba}^d] = [k_{ab}^d]^T . \tag{36}$$

The actual 6x6 partitions are formed by expanding:

$$[k^d_{aa}] \Longrightarrow \left[\begin{array}{c|c} k^d_{aa} & 0 \\ \hline 0 & 0 \end{array}\right], \text{ etc.} \tag{37}$$

4.87.1.7 Piecewise Linear Analysis Calculations (Subroutine PSRØD of Module PLA3 and Subroutine PKRØD of Module PLA4)

The additional ECPTNL and ESTNL entries are:

- ε^*_o - The previously computed strain value once removed.
- ε^* - The previously computed strain value.
- E^* - The previously computed modulus of elasticity.
- T* - The previously computed torsional moment (present in the ESTNL entry only).

All of the above values are initially zero with the exception of E*, which is initially the original modulus of elasticity present on a MAT1 card.

For both stress calculation and stiffness matrix generation, the quantities ℓ and $\{n\}$ are generated as in section 4.87.1.2.

Using $\{\Delta u^t_a\}$ and $\{\Delta u^t_b\}$, the 3x1 translational displacement vectors, calculate the increment of strain:

$$\Delta\varepsilon = \frac{1}{\ell}\{n\}^T\left[[T_a]\{\Delta u^t_a\} - [T_b]\{\Delta u^t_b\}\right], \tag{38}$$

$$\Delta\varepsilon^* = \varepsilon^* - \varepsilon^*_o . \tag{39}$$

Define the following terms:

$$\varepsilon_1 = \varepsilon^* + \Delta\varepsilon, \qquad \text{(current strain);} \tag{40}$$

$$\varepsilon_2 = \varepsilon^* + \frac{(\Delta\varepsilon + \gamma^*\Delta\varepsilon^*)(\gamma+1)}{(1+\gamma^*)} + \frac{\gamma^*(\Delta\varepsilon - \gamma^*\Delta\varepsilon^*)(\gamma+1)^2}{(1+\gamma^*)}, \tag{41}$$

(estimated next strain);

where γ is the ratio of the next load increment to the present load increment.

We calculate:

$$\sigma_1 = f(\varepsilon_1), \tag{42}$$

$$\sigma_2 = f(\varepsilon_2), \tag{43}$$

where f is the tabular stress-strain function. (When $\varepsilon^* = 0$, define $\sigma_1 = E_o\varepsilon_1$)

For stiffness matrix generation, the new material properties are:

$$E = \begin{cases} \dfrac{\sigma_2 - \sigma_1}{\varepsilon_2 - \varepsilon_1}, \text{ if } \varepsilon_2 \neq \varepsilon_1 \\ E^*, \text{ if } \varepsilon_2 = \varepsilon_1 \end{cases}; \tag{44}$$

and

$$G = \frac{E}{E_o} G_o, \tag{45}$$

where E_o and G_o are elastic moduli obtained from the MAT1 bulk data card via subroutine MAT.

For plastic element stresses and forces the values are:

$$\sigma = \sigma_1, \tag{46}$$

$$P = A\alpha_1, \tag{47}$$

$$T = \frac{JG^*}{\ell} \{n\}^T \left([T_a]\{\Delta u_a^r\} - [T_b]\{\Delta u_b^r\}\right) + T^*, \tag{48}$$

$$\tau = \frac{CT}{J} \quad (\tau = 0 \text{ if } C = 0 \text{ or } J = 0), \tag{49}$$

where

$$G^* = \frac{E^*}{E_o} G_o. \tag{50}$$

The new ESTNL and ECPTNL entries are:

$$\varepsilon^*_{on} = \varepsilon^* \quad , \tag{51}$$

$$\varepsilon^*_{n} = \varepsilon_1 \quad , \tag{52}$$

$$E^*_{n} = \frac{\sigma_2 - \sigma_1}{\varepsilon_2 - \varepsilon_1} \quad , \tag{53}$$

$$T^*_{n} = T \quad . \tag{54}$$

4.87.1.8 Coupled Mass Matrix Calculation (Subroutine MCRØD of Module SMA2)

1. The length of the element, ℓ, the normalized direction vector, {n}, and the mass of the element, m, are calculated as in Equations 1 and 2 in section 4.87.1.2 and Equation 9 in section 4.87.1.3 respectively.

2. The 3 by 3 matrix

$$[\Delta M] = \frac{m}{12}\begin{bmatrix} n_1^2 & n_1n_2 & n_1n_3 \\ n_1n_2 & n_2^2 & n_2n_3 \\ n_1n_3 & n_2n_3 & n_3^2 \end{bmatrix} \tag{55}$$

is calculated.

3. The 3 by 3 element mass matrices in basic coordinates are

$$[m_{aa}] = [m_{bb}] = \begin{bmatrix} m/2 & 0 & 0 \\ 0 & m/2 & 0 \\ 0 & 0 & m/2 \end{bmatrix} - [\Delta M] \quad , \tag{56}$$

and

$$[m_{ab}] = [m_{ba}] = [\Delta M] \quad . \tag{57}$$

4. In global coordinates the 6 by 6 mass matrix partitions are:

$$[M_{ij}] = \left[\begin{array}{c|c} T_i^T m_{ij} T_j & 0 \\ \hline 0 & 0 \end{array}\right] \quad , \tag{58}$$

for i=a or b, j = a or b and where $[T_i]$ is the global-to-basic coordinate transformation matrix for point i.

4.87.2 The BAR Element

4.87.2.1 Input Data for the BAR Element

1. The ECPT/EST entries for the BAR are:

Symbol	Descriptions
SIL_a, SIL_b	Scalar indices of grid points a and b
N_a, X_a, Y_a, Z_a N_b, X_b, Y_b, Z_b	Local coordinate system number and location in basic coordinates of the grid points
x_1, x_2, x_3	Orientation vector (see Figure 1 in section 1.3 of the User's Manual)
F	Flag for orientation vector definition
P_a, P_b	Pin flags for either end
Mat I. D.	Material property identification number
A	Cross-sectional area
I_1, I_2	Bending inertials in element coordinates about axes normal to reference planes 1 and 2 respectively
I_{12}	Cross-product bending inertia
J	Torsional Inertia
μ	Nonstructural mass per unit length
a_x, a_y, a_z b_x, b_y, b_z	Vectors defining offset distances between BAR ends and grid points (see Figure 1 in section 1.3 of the User's Manual)
K_1, K_2	Shear factors
c_1, c_2, d_1, d_2 f_1, f_2, g_1, g_2	Positions on cross section of four points for stress calculations (see section 1.3.2 of the User's Manual)
t_μ	Temperature for material properties

2. Coordinate system data

The location (X_i, Y_i, Z_i) and local coordinate system number (N_i) of each grid point (i = a or b) are used to calculate the 3 by 3 global-to-basic coordinate transformation matrices, $[T_a]$ and $[T_b]$.

3. Material data

The material identification number "Mat I.D." and t_μ are used to select the following:

E	-	Modulus of elasticity
G	-	Shear modulus
ν	-	Poisson's ratio
ρ	-	Density
α	-	Thermal expansion coefficient
T_o	-	Reference temperature
g_e	-	Structural damping ratio
σ_t	-	Stress limit, tension
σ_c	-	Stress limit, compression
σ_s	-	Stress limit, shear

4.87.2.2 Stiffness Matrix Calculation (Subroutine KBAR of Module SMA1)

1. If the orientation flag F is nonzero, transform the given vector to basic coordinates:

$$\{v_o\} = [T_a] \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} . \tag{1}$$

Otherwise,

$$\{v_o\} = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} . \tag{2}$$

2. Transfer the relative beam end locations to basic coordinates:

$$\begin{Bmatrix} \delta_{a1} \\ \delta_{a2} \\ \delta_{a3} \end{Bmatrix} = [T_a] \begin{Bmatrix} a_x \\ a_y \\ a_z \end{Bmatrix} , \tag{3}$$

$$\begin{Bmatrix} \delta_{b1} \\ \delta_{b2} \\ \delta_{b3} \end{Bmatrix} = [T_b] \begin{Bmatrix} b_x \\ b_y \\ b_z \end{Bmatrix} . \tag{4}$$

3. The center axis of the beam, defined as {i} is calculated as:

$$\{V_i\} = - \begin{Bmatrix} X_a - X_b + \delta_{a1} - \delta_{b1} \\ Y_a - Y_b + \delta_{a2} - \delta_{b2} \\ Z_a - Z_b + \delta_{a3} - \delta_{b3} \end{Bmatrix}, \tag{5}$$

$$\ell = (V_{i1}^2 + V_{i2}^2 + V_{i3}^2)^{1/2} , \tag{6}$$

$$\{i\} = \frac{1}{\ell} \{V_i\} . \tag{7}$$

4. The bending axis of the beam in plane 2 is:

$$\{k\} = \frac{\{i\} \times \{v_o\}}{|\{i\} \times \{v_o\}|} . \tag{8}$$

5. The bending axis of the beam in plane 1 is:

$$\{j\} = \{k\} \times \{i\} . \tag{9}$$

6. The 6x6 matrix for transforming element displacements in the element coordinates to basic coordinate displacements is:

$$[T_{eb}] = \left[\begin{array}{ccc:ccc} \{i\} & \{j\} & \{k\} & 0 & 0 & 0 \\ & & & \{i\} & \{j\} & \{k\} \end{array}\right] . \tag{10}$$

7. The 6 by 6 matrices for transforming global coordinate displacements to basic coordinate displacements are:

$$[C_a] = \left[\begin{array}{c:c} T_a & 0 \\ \hline 0 & T_a \end{array}\right] , \tag{11}$$

$$[C_b] = \left[\begin{array}{c|c} T_b & 0 \\ \hline 0 & T_b \end{array}\right] . \qquad (12)$$

8. The 6 by 6 matrices for transforming displacements of the grid points to displacements of the element ends are:

$$[E_a] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & a_z & -a_y \\ 0 & 1 & 0 & -a_z & 0 & a_x \\ 0 & 0 & 1 & a_y & -a_x & 0 \\ \hline & & & 1 & 0 & 0 \\ & 0 & & 0 & 1 & 0 \\ & & & 0 & 0 & 1 \end{array}\right] , \qquad (13)$$

$$[E_b] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & b_z & -b_y \\ 0 & 1 & 0 & -b_z & 0 & b_x \\ 0 & 0 & 1 & b_y & -b_x & 0 \\ \hline & & & 1 & 0 & 0 \\ & 0 & & 0 & 1 & 0 \\ & & & 0 & 0 & 1 \end{array}\right] . \qquad (14)$$

9. The 6 by 6 partitions of the element stiffness matrix in element coordinates are:

$$[K^e_{aa}] = \begin{bmatrix} \frac{AE}{\ell} & 0 & 0 & 0 & 0 & 0 \\ 0 & R_1 & \beta & 0 & -\frac{\ell}{2} & \frac{\ell}{2}R_1 \\ 0 & \beta & R_2 & 0 & -\frac{\ell}{2}R_2 & \frac{\ell}{2}\beta \\ 0 & 0 & 0 & \frac{GJ}{\ell} & 0 & 0 \\ 0 & -\frac{\ell}{2}\beta & -\frac{\ell}{2}R_2 & 0 & k_2 & -\frac{\ell^2}{3}\beta \\ 0 & \frac{\ell}{2}R_1 & \frac{\ell}{2}\beta & 0 & -\frac{\ell^2}{3}\beta & k_1 \end{bmatrix}, \quad (15)$$

$$[K^e_{ab}] = \begin{bmatrix} -\frac{AE}{\ell} & 0 & 0 & 0 & 0 & 0 \\ 0 & -R_1 & -\beta & 0 & -\frac{\ell}{2}\beta & \frac{\ell}{2}R_1 \\ 0 & -\beta & -R_2 & 0 & -\frac{\ell}{2}R_2 & \frac{\ell}{2}\beta \\ 0 & 0 & 0 & -\frac{GJ}{\ell} & 0 & 0 \\ 0 & \frac{\ell}{2}\beta & \frac{\ell}{2}R_2 & 0 & k_4 & -\frac{\ell^2}{6}\beta \\ 0 & -\frac{\ell}{2}R_1 & -\frac{\ell}{2}\beta & 0 & -\frac{\ell^2}{6}\beta & k_3 \end{bmatrix}, \quad (16)$$

$$[K^e_{ba}] = [K_{ab}]^T, \quad (17)$$

$$[K^e_{bb}] = \begin{bmatrix} \frac{AE}{\ell} & 0 & 0 & 0 & 0 & 0 \\ 0 & R_1 & \beta & 0 & \frac{\ell}{2}\beta & -\frac{\ell}{2}R_1 \\ 0 & \beta & R_2 & 0 & \frac{\ell}{2}R_2 & -\frac{\ell}{2}\beta \\ 0 & 0 & 0 & \frac{GJ}{\ell} & 0 & 0 \\ 0 & \frac{\ell}{2}\beta & \frac{\ell}{2}R_2 & 0 & k_2 & -\frac{\ell^2}{3}\beta \\ 0 & -\frac{\ell}{2}R_1 & -\frac{\ell}{2}\beta & 0 & -\frac{\ell^2}{3}\beta & k_1 \end{bmatrix} . \quad (18)$$

The terms are defined as:

If $I_{12} = 0$:

$$\beta = 0 , \quad (19)$$

$$R_1 = \frac{12EI_1}{\ell^3}\left[1 + \frac{12EI_1}{K_1AG\ell^2}\right]^{-1} , \quad (20)$$

$$R_2 = \frac{12EI_2}{\ell^3}\left[1 + \frac{12EI_2}{K_2AG\ell^2}\right]^{-1} . \quad (21)$$

Note: If $K_i A G = 0$, set $\frac{1}{K_i A G} = 0$, $i = 1$ or 2

If $I_{12} \neq 0$

$$R_1 = \frac{12EI_1}{\ell^3} , \quad (22)$$

$$R_2 = \frac{12EI_2}{\ell^3} , \quad (23)$$

$$\beta = \frac{12EI_{12}}{\ell^3} . \quad (24)$$

Note: In this case no shearing deformations are calculated.

For both cases

$$k_1 = \frac{\ell^2}{4} R_1 + \frac{EI_1}{\ell} , \tag{25}$$

$$k_2 = \frac{\ell^2}{4} R_2 + \frac{EI_2}{\ell} , \tag{26}$$

$$k_3 = \frac{\ell^2}{4} R_1 - \frac{EI_1}{\ell} , \tag{27}$$

$$k_4 = \frac{\ell^2}{4} R_2 - \frac{EI_2}{\ell} . \tag{28}$$

10. Process the end condition ("pin") data. The nonzero digits of the "pin flag" integers P_a and P_b specify the following:

$P =$

1 implies no forces are transmitted to the element in the x-direction at the pinned end
2 implies no forces are transmitted to the element in the y-direction at the pinned end
3 implies no forces are transmitted to the element in the z-direction at the pinned end
4 implies no forces are transmitted to the element in the θ_x-direction at the pinned end
5 implies no forces are transmitted to the element in the θ_y-direction at the pinned end
6 implies no forces are transmitted to the element in the θ_z-direction at the pinned end

1) Nonzero digits of the number P specify the unconnected degrees of freedom on the end of the BAR.

2) Construct the overall element matrix and perform the following operations:

a)

$$\left[\begin{array}{c|c} k^e_{aa} & k^e_{ab} \\ \hline k^e_{ba} & k^e_{bb} \end{array}\right] = \begin{bmatrix} k_{11} & k_{12} & \cdots \\ k_{21} & & \\ \vdots & & k_{12,12} \end{bmatrix} . \tag{29}$$

b) Convert the pin numbers to row numbers in the [k] matrix. If a pin number refers to end "a", it corresponds to the row number. If it refers to end "b", the row number is obtained by adding six to the pin number.

c) For each row of the [k] matrix perform the following operation to obtain the new stiffness matrix [k']

$$k'_{j\ell} = k_{j\ell} - \frac{k_{i\ell}k_{ji}}{k_{ii}} \quad \begin{cases} j = 1,\ldots 12,\ j \neq i \\ \ell = 1,\ldots 12,\ \ell \neq i \end{cases}, \tag{30}$$

and $k'_{j\ell} = 0$ for $j = i$ or $\ell = i$, where i is the row number obtained from the pin number as in b).

This operation causes the i^{th} row and column to be zero, and disconnects that degree of freedom from the matrix. Repeat for each pin index.

d) Repartition the matrix into the four original sections, carrying the zero rows and columns along.

11. The equations to convert the partitions to global coordinates are:

$$[k_{aa}] = \{T_{eb}^T C_a E_a\}^T \ [k_{aa}^e] \ \{T_{eb}^T C_a E_a\}, \tag{31}$$

$$[k_{ab}] = \{T_{eb}^T C_a E_a\}^T \ [k_{ab}^e] \{T_{eb}^T C_b E_b\}, \tag{32}$$

$$[k_{bb}] = \{T_{eb}^T C_b E_b\}^T \ [k_{bb}^e] \ \{T_{eb}^T C_b E_b\}, \tag{33}$$

$$[k_{ba}] = [k_{ab}]^T. \tag{34}$$

4.87.2.3 Lumped Mass Matrix Calculation (Subroutine MBAR of Module SMA2)

$$[M_a] = [M_b] = \begin{bmatrix} m/2 & & & 0 \\ & m/2 & & \\ & & m/2 & \\ \hline 0 & & & 0 \end{bmatrix} , \tag{35}$$

where:

$$m = \ell (\rho A + \mu) , \tag{36}$$

and:

$$[M_{aa}] = \{T_{eb}^T C_a E_a\}^T [M_a] \{T_{eb}^T C_a E_a\} , \tag{37}$$

$$[M_{bb}] = \{T_{eb}^T C_b E_b\}^T [M_b] \{T_{eb}^T C_b E_b\} , \tag{38}$$

$$[M_{ab}] = [M_{ba}] = [0] . \tag{39}$$

The equations for the generation of the "consistent" or coupled mass matrix for the BAR are given in section 4.87.2.8.

4.87.2.4 Element Load Calculation (Subroutine BAR of Module SSG1)

Form ℓ, $\{i\}$, $[C_a]$, $[C_b]$, $[E_a]$ and $[E_b]$ as in section 4.87.2.2.

Then:

$$\{P_a\} = [E_a]^T [C_a]^T \{\frac{i}{0}\} \frac{EA}{\ell} (\delta + \alpha \ell \bar{t}) , \tag{40}$$

$$\{P_b\} = -[E_b]^T [C_b]^T \{\frac{i}{0}\} \frac{EA}{\ell} (\delta + \alpha \ell \bar{t}) , \tag{41}$$

where:

$$\bar{t} = \frac{1}{2} (t_a + t_b) - T_o , \tag{42}$$

and δ is the element deformation from the EDT data block, and t_a and t_b are grid point temperatures for loading obtained from the GPTT data block.

Note: If the "1" index exists in either "pin flag", all the above loads are zero.

4.87.2.5 Element Stress Calculations (Subroutines SBAR1 and SBAR2 of Module SDR2)

The stress and force data are calculated in two phases. The first phase (subroutine SBAR1) calculates unique stress versus displacement, temperature and enforced deformation functions for each element. The second phase (subroutine SBAR2) applies the various subcase displacement vectors to produce the element forces and stresses.

Phase 1 calculations are as follows:

1. Using the algorithms given in the description of the stiffness matrix calculations for the element (section 4.87.2.2), calculate the following data:

$[T_{eb}]$ - 6x6 element coordinate transformation

$[E_a]$, $[E_b]$ - Offset transformation matrices (6x6)

$[C_a]$, $[C_b]$ - 6x6 global to basic coordinate transformations

$[K'_e]$ - 12x12 stiffness matrix in element coordinates with pin joint effects

ℓ - Length of BAR

2. Partition the stiffness matrix $[K'_e]$ saving only the upper 6x6 matrices, $[k_{aa}]$ and $[k_{ab}]$.

$$[K'_e] \Rightarrow \left[\begin{array}{c|c} k_{aa} & k_{ab} \\ \hline k_{ba} & k_{bb} \end{array}\right] . \tag{43}$$

3. The stress matrices are:

$$[S_a] = [k_{aa}][T_{eb}]^T[C_a][E_a], \tag{44}$$

$$[S_b] = [k_{ab}][T_{eb}]^T[C_b][E_b]. \tag{45}$$

4. The temperature and enforced deformation coefficients are:

$$S_t = AE\alpha, \tag{46}$$

$$S_\delta = \frac{AE}{\ell} . \tag{47}$$

Phase 2 element force calculations are as follows:

1. The static element forces are calculated by the equation:

$$\{P\} = [S_a]\{u_a\} + [S_b]\{u_b\}, \tag{48}$$

where $\{u_a\}$ and $\{u_b\}$ are the global displacements.

2. The element axial force is:

$$F_x = - [P_1 + S_t \Delta t + S_\delta \delta] , \tag{49}$$

where

$$\Delta t = \frac{t_a + t_b}{2} - T_o . \tag{50}$$

3. The element shear loads are:

$$V_1 = -P_2 , \tag{51}$$

$$V_2 = -P_3 . \tag{52}$$

4. The torque and moments are:

$$T = -P_4 , \tag{53}$$

$$M_{1a} = -P_6 , \tag{54}$$

$$M_{2a} = P_5 , \tag{55}$$

$$M_{1b} = M_{1a} - V_1 \ell , \tag{56}$$

$$M_{2b} = M_{2a} - V_2 \ell . \tag{57}$$

Phase 2 element stress calculation are as follows:

1. The stresses due to bending are:

$$k_{1a} = \frac{M_{2a} I_{12} - M_{1a} I_2}{I_1 I_2 - I_{12}^2}, \tag{58}$$

$$k_{2a} = \frac{M_{1a} I_{12} - M_{2a} I_1}{I_1 I_2 - I_{12}^2}, \tag{59}$$

$$\sigma_{ca} = k_{1a} c_1 + k_{2a} c_2, \tag{60}$$

$$\sigma_{da} = k_{1a} d_1 + k_{2a} d_2, \tag{61}$$

$$\sigma_{fa} = k_{1a} f_1 + k_{2a} f_2, \tag{62}$$

$$\sigma_{ga} = k_{1a} g_1 + k_{2a} g_2. \tag{63}$$

For σ_{cb}, σ_{db}, σ_{fb}, σ_{gb} use the above equations interchanging the subscripts for b and a.

2. The axial stress is:

$$\sigma_{ax} = \frac{F_x}{A}. \tag{64}$$

3. The maxima and minima are:

$$\sigma_{a\,max} = \sigma_{ax} + \max(\sigma_{ca}, \sigma_{da}, \sigma_{fa}, \sigma_{ga}), \tag{65}$$

$$\sigma_{b\,max} = \sigma_{ax} + \max(\sigma_{cb}, \sigma_{db}, \sigma_{fb}, \sigma_{gb}), \tag{66}$$

$$\sigma_{a\,min} = \sigma_{ax} + \min(\sigma_{ca}, \sigma_{da}, \sigma_{fa}, \sigma_{ga}), \tag{67}$$

$$\sigma_{b\,min} = \sigma_{ax} + \min(\sigma_{cb}, \sigma_{db}, \sigma_{fb}, \sigma_{gb}). \tag{68}$$

4. The margins of safety in tension, $M.S._t$, and compression, $M.S._c$, are as follows:

$$M.S._t = \begin{cases} \max\left(\frac{\sigma_t}{\sigma_{amax}}, \frac{\sigma_t}{\sigma_{bmax}}\right) - 1 & , \sigma_t > 0 \\ \text{Integer "1"} & , \sigma_t \leq 0 \end{cases} \quad (69)$$

If $M.S._t > 1.0$, then $M.S._t$ is set to integer "1".

$$M.S._c = \begin{cases} \min\left(\frac{\sigma_c}{\sigma_{amin}}, \frac{\sigma_c}{\sigma_{bmin}}\right) + 1 & , \sigma_c > 0 \\ \text{Integer "1"} & , \sigma_c \leq 0 \end{cases} \quad (70)$$

If $M.S._c > 1.0$, then $M.S._c$ is set to integer "1".

4.87.2.6 Differential Stiffness Matrix Calculation (Subroutine DBEAM of Module DSMG1)

Many of the equations used in this calculation routine are identical to the stiffness matrix and element force calculations. Refer to sections 4.87.2.2 and 4.87.2.5 for details.

1. Calculate $[T_{eb}]$, $[C_a]$, $[C_b]$, $[E_a]$, $[E_b]$ and $[K]$, the matrices used in the BAR stiffness matrix generation, section 4.87.2.2.

2. Calculate the forces in the element using the equations in section 4.87.2.5.

3. The number 2, 3, 4, 5, 6, 8, 9, 10, 11 and 12^{th} rows and columns of the 12 by 12 differential stiffness matrix are given in Figure 1. The first and seventh rows and columns are zero. The terms are identical to the element forces calculated for output (Equations 51, 52 and 54 through 57 of section 4.87.2.5) with the following notational changes:

$$M_{ay} = M_{2a} \qquad M_{by} = M_{2b}$$

$$M_{az} = M_{1a} \qquad M_{bz} = M_{1b}$$

$$V_y = V_1 \qquad V_z = V_2$$

$$[K_e^d] =$$

$\frac{6F_x}{5\ell}$	0	$\frac{M_{by}}{\ell}$	0	$-\frac{F_x}{10}$	$-\frac{6F_x}{5\ell}$	0	$\frac{M_{ay}}{\ell}$	0	$-\frac{F_x}{10}$
0	$\frac{6F_x}{5\ell}$	$\frac{M_{bz}}{\ell}$	$\frac{F_x}{10}$	0	0	$-\frac{6F_x}{5\ell}$	$\frac{M_{az}}{\ell}$	$\frac{F_x}{10}$	0
$\frac{M_{by}}{\ell}$	$\frac{M_{bz}}{\ell}$	$\frac{JF_x}{\ell^2 A}$	$-\frac{\ell V_y}{6}$	$-\frac{\ell V_z}{6}$	$-\frac{M_{by}}{\ell}$	$-\frac{M_{hz}}{\ell}$	$-\frac{JF_x}{\ell^2 A}$	$\frac{\ell V_y}{6}$	$\frac{\ell V_z}{6}$
0	$\frac{F_x}{10}$	$-\frac{\ell V_y}{6}$	$\frac{2\ell}{15}F_x$	0	0	$-\frac{F_x}{10}$	$\frac{\ell V_y}{6}$	$-\frac{\ell F_x}{30}$	0
$-\frac{F_x}{10}$	0	$-\frac{\ell V_z}{6}$	0	$-\frac{2\ell}{15}F_x$	$\frac{F_x}{10}$	0	$\frac{\ell V_z}{6}$	0	$-\frac{\ell F_x}{30}$
$-\frac{6F_x}{5}$	0	$-\frac{M_{by}}{\ell}$	0	$\frac{F_x}{10}$	$\frac{6F_x}{5\ell}$	0	$-\frac{M_{ay}}{\ell}$	0	$\frac{F_x}{10}$
0	$-\frac{6F_x}{\ell}$	$-\frac{M_{bz}}{\ell}$	$-\frac{F_x}{10}$	0	0	$\frac{6F_x}{5\ell}$	$-\frac{M_{az}}{\ell}$	$-\frac{F_x}{10}$	0
$\frac{M_{ay}}{\ell}$	$\frac{M_{az}}{\ell}$	$-\frac{JF_x}{\ell^2 A}$	$\frac{\ell V_y}{6}$	$\frac{\ell V_z}{6}$	$-\frac{M_{ay}}{\ell}$	$-\frac{M_{az}}{\ell}$	$\frac{JF_x}{\ell^2 A}$	$-\frac{\ell V_y}{6}$	$-\frac{\ell V_z}{6}$
0	$\frac{F_x}{10}$	$\frac{\ell V_y}{6}$	$-\frac{\ell F_x}{30}$	0	0	$-\frac{F_x}{10}$	$-\frac{\ell V_y}{6}$	$\frac{2\ell F_x}{15}$	0
$-\frac{F_x}{10}$	0	$\frac{\ell V_z}{6}$	0	$-\frac{\ell F_x}{30}$	$\frac{F_x}{10}$	0	$-\frac{\ell V_z}{6}$	0	$\frac{2\ell F_x}{15}$

Figure 1. - Differential stiffness matrix for a BAR element, rows and columns 1 and 7 deleted.

4. The effects of "pin joints" are added by applying the elastic stiffness constraints. The elastic stiffness matrix, $[K^e]$, with no pin joints is equivalent to the matrix [k] in Equation 29. If coordinate number i is released by a pin flag the differential stiffness matrix must be modified as follows:

a. If $i \neq j$ and $\ell \neq j$, $i = 1, 2, \ldots, 12$ and $\ell = 1, 2, \ldots, 12$:

$$(K^{d*}_{i\ell})_m = (K^{d*}_{i\ell})_{m-1} - \frac{K^e_{\ell j}\,(K^{d*}_{ji})_{m-1}}{K^e_{jj}} - \frac{K^e_{ji}\,(K^{d*}_{\ell j})_{m-1}}{K^e_{jj}} + \frac{K^e_{\ell j}\,K^e_{ji} + (K^{d*}_{jj})_{m-1}}{(K^e_{jj})^2} \quad , \tag{71}$$

where m is the (row) index of the pin joint number, $1 \leq m \leq 12$. For $m = 0$, define

$$(K^{d*}_{i\ell})_o = (K^d_{i\ell}) \quad . \tag{72}$$

b. If i or $j = \ell$

$$K^{d*}_{ij} = 0 \qquad i = 1, \ldots, 12 \quad , \tag{73}$$

$$K^{d*}_{j\ell} = 0 \qquad j = 1, \ldots, 12 \quad . \tag{74}$$

5. The 12 by 12 matrix $[K^{d*}]$ is now partitioned into 6 by 6 matrices related to each grid point

$$[K^{d*}] = \left[\begin{array}{c|c} K^{d*}_{aa} & K^{d*}_{ab} \\ \hline K^{d*}_{ba} & K^{d*}_{bb} \end{array}\right] \quad . \tag{75}$$

6. If point "p" is the pivot point (p = a or b), the matrices generated in global coordinates are:

$$[K^d_{pa}] = ([T_{eb}]^T[C_p][E_p])^T\,[K^{d*}_{pa}]\,([T_{eb}][C_a][E_a]) \quad , \tag{76}$$

$$[K^d_{pb}] = ([T_{eb}]^T[C_p][E_p])^T\,[K^{d*}_{pb}]\,([T_{eb}]^T[C_b][E_b]) \quad . \tag{77}$$

4.87.2.7 Piecewise Linear Analysis Calculations (Subroutine PSBAR of Module PLA3 and Subroutine PKBAR of Module PLA4)

The additional ECPTNL and ESTNL data block entries for a BAR element are:

ε_o^* - The previously computed axial strain value once removed.

ε^* - The previously computed axial strain value.

E^* - The previously computed modulus of elasticity.

V_1^*, V_2^*, T^*, M_{1a}^*, M_{2a}^* - The previously computed element forces and moments.

All of the above values are initially zero with the exception of E^*, which is initially the original modulus of elasticity present on a MAT1 bulk data card.

For both stress (subroutine PSBAR) and stiffness matrix (subroutine PKBAR) calculations, the following data are generated:

ℓ, $[T_{eb}]$, $[C_a]$, $[C_b]$, $[E_a]$, $[E_b]$, $[k^e_{aa}]$, and $[k^e_{ab}]$ as in Equations 6, 10, 11, 12, 13, 14 and 30 in section 4.87.2.1. Note that: a) $[k^e_{aa}]$ and $[k^e_{ab}]$ are the partitions of the stiffness matrix with pin joint effects taken into account; b) for stress calculations, E^* is used to compute $[k^e_{aa}]$ and $[k^e_{ab}]$; and c) for stiffness matrix calculations E_1 (see Equation 84 below) is used to compute $[k^e_{aa}]$ and $[k^e_{ab}]$.

Using the incremental displacement vectors, $\{\Delta u_a\}$ and $\{\Delta u_b\}$, calculate the incremental strain:

$$\Delta\varepsilon = \frac{1}{\ell}\{T_{eb1}\}^T\left[[C_b][E_b]\{\Delta u_b\} - [C_a][E_a]\{\Delta u_a\}\right], \qquad (78)$$

where $\{T_{eb1}\}$ is the first column of $[T_{eb}]$. If coordinate "1" of either P_a or P_b (the pin flags) is "on", the element is treated as linear. This determination is made in the Piecewise Linear

Analysis pre-processor module, PLA1.

Calculate the extensional strains:

$$\Delta\varepsilon^* = \varepsilon^* - \varepsilon_o^* \ , \tag{79}$$

$$\varepsilon_1 = \varepsilon^* + \Delta\varepsilon \ , \tag{80}$$

$$\varepsilon_2 = \varepsilon^* + \frac{(\Delta\varepsilon + \gamma^{*2}\Delta\varepsilon^*)(\gamma + 1)}{(1 + \gamma^*)} + \frac{\gamma^*(\Delta\varepsilon - \gamma^*\Delta\varepsilon^*)(\gamma + 1)^2}{(1 + \gamma^*)} , \tag{81}$$

where γ is the ratio of the next load increment to the present load increment.

The stresses

$$\sigma_1 = f(\varepsilon_1), \tag{82}$$

$$\sigma_2 = f(\varepsilon_2), \tag{83}$$

are computed, where f is the tabular stress-strain function. (When $\varepsilon^* = 0$, define $\sigma_1 = E_o\varepsilon_1$, where E_o is the modulus of elasticity on the MAT1 card)

For stiffness matrix generation the new material properties are:

$$E_1 = \begin{cases} \dfrac{\sigma_2 - \sigma_1}{\varepsilon_2 - \varepsilon_1} \ , & \text{if } \varepsilon_2 \neq \varepsilon_1 \\ \\ E^* \ , & \text{if } \varepsilon_2 = \varepsilon_1 \end{cases} ; \tag{84}$$

and

$$G_1 = \frac{E_1}{E_0} G_0 \ , \tag{85}$$

where E_0 and G_0 are elastic moduli obtained from the MAT1 bulk data card via subroutine MAT. Note that E_1 is calculated in PSBAR only to predict the next value of E, i.e., to update the ESTNL entry.

For plastic element stresses and forces, the values are calculated in a fashion similar to that found in the phase 2 subroutine, SBAR2, (see section 4.87.2.5) of the SDR2 module. They are:

$$\{\Delta P\} = [k_{aa}][T_{eb}]^T[C_a][E_a]\{\Delta u_a\} + [k_{ab}][T_{eb}]^T[C_b][E_b]\{\Delta u_b\}, \quad (86)$$

$$F_x = A\sigma_1, \quad (87)$$

$$V_1 = -\Delta P_2 + V_1^*, \quad (88)$$

$$V_2 = -\Delta P_3 + V_2^*, \quad (89)$$

$$T = -\Delta P_4 + T^*, \quad (90)$$

$$M_{1a} = -\Delta P_6 + M_{1a}^*, \quad (91)$$

$$M_{2a} = \Delta P_5 + M_{2a}^*, \quad (92)$$

$$M_{1b} = M_{1a} - V_1\ell, \quad (93)$$

$$M_{2b} = M_{2a} - V_2\ell. \quad (94)$$

The stresses due to bending, the axial stress, the minimum and maximum stresses, and the margins of safety are computed as in Equations 58 through 70.

The new ESTNL and ECPTNL entries are:

$$\varepsilon_o^* = \varepsilon^*, \quad (95)$$

$$\varepsilon^* = \varepsilon_1, \quad (96)$$

$$E^* = E_1, \quad (97)$$

$$V_1^* = V_1 , \tag{98}$$

$$V_2^* = V_2 , \tag{99}$$

$$T^* = T , \tag{100}$$

$$M_{1a}^* = M_{1a} , \tag{101}$$

$$M_{2a}^* = M_{2a} . \tag{102}$$

4.87.2.8 "Consistent" Mass Matrix Calculation (Subroutine MCBAR of Module SMA2)

1. Generate the 12 by 12 matrix:

$$[M^e] = \frac{m}{420}\left[\begin{array}{cccccc|cccccc} 175 & 0 & 0 & 0 & 0 & 0 & 35 & 0 & 0 & 0 & 0 & 0 \\ & 156 & 0 & 0 & 0 & 22\ell & 0 & 54 & 0 & 0 & 0 & -13\ell \\ & & 156 & 0 & -22\ell & 0 & 0 & 0 & 54 & 0 & 13\ell & 0 \\ & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & 4\ell^2 & 0 & 0 & 0 & -13\ell & 0 & -3\ell^2 & 0 \\ & & & & & 4\ell^2 & 0 & 13\ell & 0 & 0 & 0 & -3\ell^2 \\ \hline & & & \text{SYM} & & & 175 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & 156 & 0 & 0 & 0 & -22\ell \\ & & & & & & & & 156 & 0 & 22\ell & 0 \\ & & & & & & & & & 0 & 0 & 0 \\ & & & & & & & & & & 4\ell^2 & 0 \\ & & & & & & & & & & & 4\ell^2 \end{array}\right], \quad (103)$$

where

$$m = (\rho A + \mu)\ell \quad . \quad (104)$$

2. If "pin joints" exist (P_a or P_b nonzero), generate the "unpinned" 12 by 12 stiffness matrix in element coordinates, $[K^e]$, as in Equation 29, section 4.87.2.2.

For each pin joint of index j, perform the operations for i = 1, ..., 12 and ℓ = 1, ..., 12:

$$M^p_{i\ell} = M^e_{i\ell} - \frac{K^e_{\ell j} M^e_{ji}}{K^e_{jj}} - \frac{K^e_{ji} M^e_{\ell j}}{K^e_{jj}} + \frac{K^e_{\ell j} K^e_{ji} M^e_{jj}}{(K^e_{jj})^2} \quad . \quad (105)$$

After each pin joint j operation, replace $[M^e]$ by the "pinned" matrix $[M^p]$.

4. Partition the matrix into 6 by 6 submatrices:

$$[M] = \left[\begin{array}{c|c} M_{aa} & M_{ab} \\ \hline M_{ba} & M_{bb} \end{array}\right] \quad . \tag{106}$$

5. The matrices are converted to global coordinates by the equation:

$$[M^g_{ij}] = \left| [T_{eb}]^T [C_i][E_i] \right|^T [M_{ij}] \left| [T_{eb}]^T [C_j][E_j] \right| \quad , \tag{107}$$

where i is the pivot point (a or b) and j is used twice (for both a and b).

4.87.3 The SHEAR Panel and TWIST Panel Elements

4.87.3.1 Input Data for SHEAR and TWIST Panels

1. The ECPT/EST entries for shear (SHEAR) and twist (TWIST) panel elements are:

Symbol	Description
SIL_i, i=1,2,3,4	Scalar indices for the connected points
N_i, X_i, Y_i, Z_i } i = 1,2,3,4	Local coordinate system number and basic coordinate location for each of the connected points.
Mat I.D.	Material identification number
t	Panel thickness
μ	Nonstructural mass per unit area
t_μ	Temperature for material properties

2. Coordinate system data

Using N_i, X_i, Y_i, Z_i, i = 1,2,3,4 the program constructs $[T_i]$, i = 1,2,3,4, the 3 by 3 global-to-basic transformation matrix for each point.

3. Material data

MAT I.D. and t_μ are used, by utility routine MAT, to produce the following terms from the MPT and DIT data blocks:

Symbol	Description
E	Modulus of elasticity
G	Shear modulus
ν	Poisson's ratio
ρ	Density
α	Thermal expansion coefficient
T_o	Reference temperature
g_e	Structural damping coefficient
σ_t, σ_c, σ_b	Stress limits

4.87.3.2 Definition of Element Geometry

A mean plane is defined as parallel to the two diagonal lines and halfway between them. The projections of the points at the corners of the element onto the plane and the normal to the plane define the element coordinate system. Using standard vector algebra, the steps are:

1. Define:

$$\{V_{01}\} = \begin{Bmatrix} X_1 \\ Y_1 \\ Z_1 \end{Bmatrix}, \quad \{V_{02}\} = \begin{Bmatrix} X_2 \\ Y_2 \\ Z_2 \end{Bmatrix}, \text{ etc.} \tag{1}$$

2. Define diagonal vectors:

$$\{v_{d1}\} = \{V_{03}\} - \{V_{01}\}, \tag{2}$$

$$\{v_{d2}\} = \{V_{04}\} - \{V_{02}\}. \tag{3}$$

3. Define normal vector (x denotes cross product):

$$\{k_n\} = \{v_{d1}\} \times \{v_{d2}\}, \tag{4}$$

$$\{k\} = \frac{\{k_n\}}{|\{k_n\}|}, \tag{5}$$

$$A = \frac{1}{2} |\{k_n\}| \quad \text{(the projected area of the element).} \tag{6}$$

4. Define the vectors along the side of the element (see Figures 2 and 3)

$$\{v_{12}\} = \{V_{02}\} - \{V_{01}\}, \tag{7}$$

$$\{v_{41}\} = \{V_{01}\} - \{V_{04}\}. \tag{8}$$

5. Define transformation matrix $[T_e]$, which transforms element coordinate to basic coordinates, using unit vectors:

$$\{v_{12}^p\} = \{v_{12}\} - (\{v_{12}\}^T \{k\})\{k\} \quad , \tag{9}$$

$$\{i\} = \frac{\{v_{12}^p\}}{|\{v_{12}^p\}|} \quad , \tag{10}$$

$$\{j\} = \{k\} \times \{i\} \quad , \tag{11}$$

$$[T_e] = \begin{bmatrix} i_1 & j_1 \\ i_2 & j_2 \\ i_3 & j_3 \end{bmatrix} \quad . \tag{12}$$

6. Transform the four corner point of the element from basic coordinates to the element system:

$$\{r_1\} = \begin{Bmatrix} x_1 \\ y_1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad , \tag{13}$$

$$\{r_2\} = \begin{Bmatrix} x_2 \\ y_2 \end{Bmatrix} = \begin{Bmatrix} |v_{12}^p| \\ 0 \end{Bmatrix} , \tag{14}$$

$$\{r_3\} = \begin{Bmatrix} x_3 \\ y_3 \end{Bmatrix} = [T_e]^T \{v_{d1}\} \quad , \tag{15}$$

$$\{r_4\} = \begin{Bmatrix} x_4 \\ y_4 \end{Bmatrix} = - [T_e]^T \{v_{41}\} \quad . \tag{16}$$

The four corners of the element are now projected onto the mean plane.

7. The following conditions should be met. Otherwise, the interior angle at the indicated point is not valid.

$$y_3 > 0 \qquad \text{(If not, the interior angle at point 2 > 180}^{\circ}\text{)} \quad , \quad (17)$$

$$x_3 > \frac{y_3}{y_4} x_4 \qquad \text{(If not, the interior angle at point 4 > 180}^{\circ}\text{)} \quad , \quad (18)$$

$$y_4 > 0 \qquad \text{(If not, the interior angle at point 1 > 180}^{\circ}\text{)} \quad , \quad (19)$$

$$x_4 < x_2 - (x_2 - x_3)\frac{y_4}{y_3} \qquad \text{(If not, the interior angle at point 3 > 180}^{\circ}\text{)} \quad . \quad (20)$$

4.87.3.3 Coefficient Generation

The shape of the panel may be a parallelogram, a trapezoid, or a general quadrilateral, and the equations will be different for each case. The slopes of the opposite sides are checked for parallel effects, and the correct routine is used for each possibility.

1. Check for parallel effects:

If

$$\left| \frac{y_3 - y_4}{x_3 - x_4} \right| < \varepsilon, \qquad (21)$$

sides 1 and 3 of the panel are parallel ($\varepsilon = 10^{-6}$).

If

$$\left| \frac{y_4(x_3 - x_2) - y_3 x_4}{x_4(x_3 - x_2)} \right| < \varepsilon, \qquad (22)$$

sides 2 and 4 are parallel. If both terms are less than ε, (i.e., the panel is a parallelogram), go to step (4). If both terms are greater than ε, go to step (5). If the one pair of parallel sides is 1 and 3, go to step (2); if the one pair of parallel sides is 2 and 4, go to step (3).

2. In this case the line connecting points 3 and 4 is approximately parallel to the line connecting points 1 and 2. The equations are:

$$y_p = \frac{x_2 y_3 y_4}{y_3 x_4 - y_4 (x_3 - x_2)} , \tag{23}$$

$$p_i = y_p - y_i \quad (i = 1,2,3,4) , \tag{24}$$

$$x_p = \frac{x_2 y_3 x_4}{y_3 x_4 - y_4 (x_3 - x_2)} , \tag{25}$$

$$a = \left(\frac{x_2 - x_p}{y_p} \right) , \tag{26}$$

$$c = \left(\frac{x_1 - x_p}{y_p} \right) , \tag{27}$$

$$Z = \frac{p_1 p_2}{p_3 p_4} \frac{A}{2Gt} \left\{ 1 + \frac{2}{3(1+\nu)} (a^2 + ac + c^2) \right\} . \tag{28}$$

3. In this case the line connecting points 1 and 4 is approximately parallel to the line connecting points 2 and 3. The equations are:

$$d = -\frac{1}{2} \left[\frac{x_4}{y_4} + \frac{x_3 - x_2}{y_3} \right] , \tag{29}$$

$$x_q = x_4 - \frac{x_3 - x_4}{y_3 - y_4} y_4 , \tag{30}$$

$$p_i = [(x_q - x_i) - y_i d] \frac{1}{\sqrt{1 + d^2}} \quad (i = 1,2,3,4) \quad , \tag{31}$$

$$b = \frac{(x_q - x_4)d + y_4}{(x_q - x_4) - y_4 d} \quad , \tag{32}$$

$$Z = \frac{p_1 p_2}{p_3 p_4} \frac{A}{2Gt} \{1 + \frac{2}{3(1+\nu)} (b^2 + bd + d^2)\}. \tag{33}$$

4. In this case the panel approximates a parallelogram. The equations to solve are:

$$p_i = 1, \quad (i = 1,2,3,4), \tag{34}$$

$$d = -\frac{1}{2}\left(\frac{x_4}{y_4} + \frac{x_3 - x_2}{y_3} + \frac{y_3 - y_4}{x_3 - x_4}\right) \quad , \tag{35}$$

$$Z = \frac{A}{2Gt} (1 + \frac{2d^2}{1+\nu}). \tag{36}$$

5. In this case no parallel effects exist. The equations are:

$$x_q = x_4 - \frac{(x_3 - x_4)}{(y_3 - y_4)} y_4 \quad , \tag{37}$$

$$x_p = \frac{x_2 x_4 y_3}{y_3 x_4 - y_4 (x_3 - x_2)} , \tag{38}$$

$$y_p = \frac{x_2 y_3 y_4}{y_3 x_4 - y_4 (x_3 - x_2)} , \tag{39}$$

$$\ell = \sqrt{(x_q - x_p)^2 + y_p^2} , \tag{40}$$

$$d = \frac{x_q - x_p}{y_p} , \tag{41}$$

$$p_i = \frac{y_p}{\ell} [(x_q - x_i) - y_i d] \quad (i = 1,2,3,4) , \tag{42}$$

$$c = \frac{\ell}{p_1} - d , \tag{43}$$

$$b = \frac{\ell}{p_4} - c, \tag{44}$$

$$a = \frac{\ell}{p_2} - d. \tag{45}$$

Let:

$$F = \frac{p_1p_2p_3p_4}{2\ell^2}\{[(a+b) + \frac{2}{3}(a^3+b^3) + \frac{1}{5}(a^5+b^5)]\log_e|a+b|$$

$$+ [(c+d) + \frac{2}{3}(c^3+d^3) + \frac{1}{5}(c^5+d^5)]\log_e|c+d|$$

$$- [(b+c) + \frac{2}{3}(b^3+c^3) + \frac{1}{5}(b^5+c^5)]\log_e|b+c| \qquad (46)$$

$$- [(d+a) + \frac{2}{3}(d^3+a^3) + \frac{1}{5}(d^5+a^5)]\log_e|d+a|$$

$$+ \frac{1}{10}[(a^2-c^2)(b^3-d^3) + (b^2-d^2)(a^3-c^3)]$$

$$- \frac{1}{5}[(a-c)(b^4-d^4) + (b-d)(a^4-c^4)]\} .$$

Then:

$$Z = \frac{p_1p_2}{p_3p_4}\frac{1}{2Gt}\left\{A + \frac{4}{1+\nu}[F - \frac{2}{3}A]\right\} . \qquad (47)$$

4.87.3.4 Stiffness Matrix Formulation For a SHEAR Panel (Subroutine KPANEL of Module SMA1)

1. Calculate the lengths of the diagonals:

$$\ell_{13} = \sqrt{x_3^2 + y_3^2} \quad , \tag{48}$$

$$\ell_{24} = \sqrt{(x_4-x_2)^2 + y_4^2} \quad , \tag{49}$$

2. Calculate the unit vectors along the diagonals:

$$u_1 = u_3 = \frac{x_3}{\ell_{13}} \quad , \tag{50}$$

$$v_1 = v_3 = \frac{y_3}{\ell_{13}} \quad , \tag{51}$$

$$u_2 = u_4 = \frac{x_4-x_2}{\ell_{24}} \quad , \tag{52}$$

$$v_2 = v_4 = \frac{y_4}{\ell_{24}} \quad . \tag{53}$$

3. The loads along the diagonals in terms of the average shear stress along side 1 are:

$$A_1 = - \frac{x_2 y_4 \ell_{13}}{2(x_4 y_3 - x_3 y_4)} \quad , \tag{54}$$

$$A_2 = \frac{x_2 y_3 \ell_{24}}{2(x_4 y_3 - x_3 y_4 - x_2(y_3 - y_4))} , \tag{55}$$

$$A_3 = -A_1 , \tag{56}$$

$$A_4 = -A_2 . \tag{57}$$

4. The loads at grid point i in terms of the displacements at grid point j may be expressed in terms of a (3x3) matrix $[k_{ij}]$ where

$$[k_{ij}] = \frac{A_i A_j}{2Z} [T_i]^T [T_e] \begin{Bmatrix} u_i \\ v_i \end{Bmatrix} \{u_j v_j\}^T [T_e]^T [T_j] , \tag{58}$$

$$i = 1, 2, 3, 4 ,$$

$$j = 1, 2, \ldots i .$$

5. The 3x3 matrices are related only to deflections and forces. The terms in the 6x6 matrices, $[K_{ij}]$, corresponding to rotations are zero. Expand the matrices to 6x6:

$$[K_{ij}] = \left[\begin{array}{c|c} k_{ij} & 0 \\ \hline 0 & 0 \end{array}\right] . \tag{59}$$

The element structural damping matrix is equal to g_e, the structural damping coefficient, multiplied by the stiffness matrix, $[K_{ij}]$.

4.87.3.5 TWIST Element Stiffness Matrix Generation (Subroutine KPANEL of Module SMA1)

The following data for the SHEAR panel element are used for generation of the element stiffness matrix for the TWIST panel.

$[T_e]$	The 3x2 transformation matrix (Equation 12).
x_2, x_3, x_4, y_3, y_4	The locations of the corners in the element system (Equations 14, 15 and 16).
Z	The energy coefficient (Equation 28, 33, 36, or 47).
$u_1 u_2 u_3 u_4$ $v_1 v_2 v_3 v_4$	Unit vector coefficients at the corners (Equations 50 through 53)
A_1, A_2, A_3, A_4	Load coefficients for the corners (Equations 54 through 57)
$[T_1], [T_2], [T_3], [T_4]$	3x3 global-to-basic transformation matrices

1. Generate the three by three matrices relating the moments at point i to the rotations at point j:

$$[q_{ij}] = \frac{A_i A_j t^2}{24Z} [T_i]^T [T_e] \begin{Bmatrix} -v_i \\ u_i \end{Bmatrix} \{-v_j u_j\}^T [T_e]^T [T_j] . \tag{60}$$

These are generated only for one point i (the pivot point) and j = 1,2,3 and 4.

2. The 3x3 matrices $[q_{ij}]$ are expanded to 6x6 matrices $[K_{ij}]$ having zeros in the translational displacement rows and columns.

$$[K_{ij}] = \begin{bmatrix} 0 & 0 \\ \hline 0 & q_{ij} \end{bmatrix} . \tag{61}$$

4.87.3.6 Mass Matrix Generation (Subroutine MASSTQ of Module SMA2)

The mass at each point is determined by cutting the quadrilateral into four overlapping triangles. Each triangle is defined by three of the four points as follows:

Triangle No.	Connected Points		
K	j1	j2	13
I	4	1	2
II	1	2	3
III	2	3	4
IV	3	4	1

The area of each triangle is determined by the equation :

$$A_K = \frac{1}{2}\left|(\{V_{oj2}\}-\{V_{oj1}\}) \times (\{V_{oj3}\}-\{V_{oj1}\})\right| , \tag{62}$$

where $\{V_{oj1}\}$ is the location vector of the first point defining the triangle, $\{V_{oj2}\}$ the second point, and $\{V_{oj3}\}$ the third point.

The mass of each triangle is divided equally among its connected points. The mass at each point is :

$$m_1 = \frac{(\mu + \rho t)}{3}(A_4 + A_1 + A_2) , \tag{63}$$

$$m_2 = \frac{(\mu + \rho t)}{3}(A_1 + A_2 + A_3) , \tag{64}$$

$$m_3 = \frac{(\mu + \rho t)}{3}(A_2 + A_3 + A_4), \tag{65}$$

$$m_4 = \frac{(\mu + \rho t)}{3}(A_3 + A_4 + A_1). \tag{66}$$

For each point a six by six diagonal mass matrix is constructed. The matrix is:

$$[M_{ii}] = \begin{bmatrix} m_i & & & & & \\ & m_i & & & 0 & \\ & & m_i & & & \\ & & & 0 & & \\ & 0 & & & 0 & \\ & & & & & 0 \end{bmatrix} . \qquad (67)$$

4.87.3.7 SHEAR Element Stress and Force Calculations (Subroutines SPANL1 and SPANL2 of Module SDR2)

The stress and force calculations are performed in two phases: phase 1 in SPANL1; phase 2 in SPANL2.

PHASE 1

1. Calculate the 1 by 3 matrices $[S_i]$, i = 1, 2, 3, 4:

$$[S_i] = -\frac{A_i}{2Zt} \{u_i \vdots v_i\} [T_e]^T [T_i], \qquad (68)$$

where A_i, Z, t, u_i, v_i, $[T_e]$ and $[T_i]$ are as given in sections 4.87.3.2 and 4.87.3.3.

2. The [S] terms and the following parameters:

$$A_1, A_2, t, \frac{P_2}{P_1}, \frac{P_1P_2}{P_3^2}, \frac{P_1P_2}{P_4^2},$$

are saved on a scratch file for phase 2 calculations. p_i, where i = 1, 2, 3 and 4 are calculated using the equations in section 4.87.3.3.

PHASE 2

1. The average stress along side 1 is:

$$\bar{s}_1 = \sum_{i=1}^{4} [S_i]\{u_i^t\}. \tag{69}$$

$\{u_i^t\}$ are the translational vectors where:

$$\{u_{gi}\} \Longrightarrow \begin{Bmatrix} u_i^t \\ \hline u_i^r \end{Bmatrix} . \tag{70}$$

2. The stresses on the corners are:

$$\tau_1 = \frac{p_2}{p_1} \bar{s}_1 , \tag{71}$$

$$\tau_2 = \frac{p_1}{p_2} \bar{s}_1 , \tag{72}$$

$$\tau_3 = \frac{p_1 p_2}{p_3^2} \bar{s}_1 , \tag{73}$$

$$\tau_4 = \frac{p_1 p_2}{p_4^2} \bar{s}_1 . \tag{74}$$

3. The average and maximum stresses are defined as:

$$\tau_{avg} = \frac{1}{4} (\tau_1 + \tau_2 + \tau_3 + \tau_4), \tag{75}$$

$$\tau_{max} = \max (|\tau_1|, |\tau_2|, |\tau_3|, |\tau_4|). \tag{76}$$

4. The margin of safety in shear is defined by :

$$M.S._s = \begin{cases} \dfrac{\sigma_s}{\tau_{max}} - 1 \ , \ \text{if } \sigma_s > 0 \\ \text{Integer "1", if } \sigma_s \leq 0 \end{cases} . \tag{77}$$

5. The net loads on the corners in the diagonal direction are :

$$P_{13} = A_1 \bar{s}_1 t \ , \tag{78}$$

$$P_{24} = A_2 \bar{s}_1 t \ . \tag{79}$$

4.87.3.8 TWIST Element Stress and Force Calculations (Subroutines SPANL1 and SPANL2 of Module SDR2)

The stress and force calculations are performed in two phases, as for the SHEAR panel element.

PHASE 1

1. Calculate for i = 1, 2, 3, 4

$$[S_i] = -\frac{A_i}{4Z} \{-v_i \vdots u_i\} [T_e]^T [T_i]. \tag{80}$$

2. The $[S_i]$ terms and the following data:

$$A_1, A_2, t, \frac{p_2}{p_1}, \frac{p_1 p_2}{p_3^2}, \frac{p_1 p_2}{p_4^2} \ .$$

are saved on a scratch file for phase 2 calculations.

PHASE 2

1. The mean outer fibre shear stress along side 1 is:

$$\tau_1 = \sum_{i=1}^{4} [S_i] \{u_i^r\}, \tag{81}$$

where $\{u_i^r\}$ are the three rotational displacements:

$$\{u_{gi}\} \Rightarrow \begin{Bmatrix} u_i^t \\ \hline u_i^r \end{Bmatrix} .$$

2. The stresses are :

$$\sigma_1 = \frac{p_2}{p_1} \tau_1, \tag{82}$$

$$\sigma_2 = \frac{\tau_1 p_1}{p_2}, \tag{83}$$

$$\sigma_3 = \frac{p_1 p_2}{p_3^2} \tau_1, \tag{84}$$

$$\sigma_4 = \frac{p_1 p_2}{p_4^2} \tau_1, \tag{85}$$

$$\sigma_{avg} = \frac{1}{4} (\sigma_1 + \sigma_2 + \sigma_3 + \sigma_4), \tag{86}$$

$$\sigma_{max} = \max(|\sigma_1|, |\sigma_2|, |\sigma_3|, |\sigma_4|). \tag{87}$$

3. The margin of safety in shear is defined by :

$$M.S._s = \begin{cases} \dfrac{\sigma_s}{\sigma_{max}} - 1 , & \sigma_s > 0 \\ \text{Integer "1"}, & \sigma_s \leq 0 \end{cases} . \tag{88}$$

4. The moments are :

$$M_{13} = \frac{A_1 t^2}{6} \tau_1, \tag{89}$$

$$M_{24} = \frac{A_2 t^2}{6} \tau_1 \quad . \tag{90}$$

4.87.3.9 SHEAR Panel Differential Stiffness Calculations (Subroutine DSHEAR of Module DSMG1)

1. Data

The data necessary for analysis are included in the ECPT, CSTM, MPT and UGV data blocks. The following data are generated as in sections 4.87.3.2, 4.87.3.3, and 4.87.3.4 .

a. $[T_i]$, i = 1,2,3,4, the 3x3 transformation matrices between global and basic coordinates, at the four corners of the shear panel.

b. $[T_e]$, the 3x2 transformations between basic and element coordinates.

c. {k}, the unit vector normal to the plane in basic coordinates.

d. u_i, v_i, i = 1,2,3,4, the unit vectors along the diagonals in element coordinates.

e. A_i, i = 1,2,3,4, the load coefficients for the corners.

f. Z, the energy coefficient for the panel.

g. ℓ_{13} and ℓ_{24}, the lengths of the diagonals.

2. Algorithm

a. The load in the diagonal between points 1 and 3 is :

$$F_{13} = -\frac{A_1}{2Z} \sum_{i=1}^{4} A_i \{u_i \vdots v_i\}^T [T_e]^T [T_i] \begin{Bmatrix} x_{1i} \\ x_{2i} \\ x_{3i} \end{Bmatrix} . \tag{91}$$

where $\{x_i\}$ is the vector of the three translations in global coordinates for point (i).

b. The load in the other diagonal is :

$$F_{24} = \frac{A_2}{A_1} F_{13} \cdot \tag{92}$$

c. Construct a perpendicular, which is defined in basic coordinates, to each diagonal vector in the plane of the panel.

$$\{j_1\} = [T_e] \begin{Bmatrix} -v_1 \\ u_1 \end{Bmatrix}, \tag{93}$$

$$\{j_2\} = [T_e] \begin{Bmatrix} -v_2 \\ u_2 \end{Bmatrix}. \tag{94}$$

d. The nonzero partitions of the overall differential stiffness matrix for the displacements are:

$$[k^d_{11}] = F'_{13}[T_1]^T \left[\{j_1\}\{j_1\}^T + \{k\}\{k\}^T\right][T_1], \tag{95}$$

$$[k^d_{13}] = -F'_{13}[T_1]^T \left[\{j_1\}\{j_1\}^T + \{k\}\{k\}^T\right][T_3], \tag{96}$$

$$[k^d_{33}] = F'_{13}[T_3]^T \left[\{j_1\}\{j_1\}^T + \{k\}\{k\}^T\right][T_3], \tag{97}$$

$$[k^d_{31}] = [k^d_{13}]^T, \tag{98}$$

$$[k^d_{22}] = F'_{24}[T_2]^T \left[\{j_2\}\{j_2\}^T + \{k\}\{k\}^T\right][T_2], \tag{99}$$

$$[k^d_{24}] = -F'_{24}[T_2]^T \left[\{j_2\}\{j_2\}^T + \{k\}\{k\}^T\right][T_4], \tag{100}$$

$$[k^d_{44}] = F'_{24}[T_4]^T \left[\{j_2\}\{j_2\}^T + \{k\}\{k\}^T\right] [T_4] , \tag{101}$$

$$[k^d_{42}] = [k^d_{24}]^T , \tag{102}$$

where :

$$F'_{13} = \frac{F_{13}}{\ell_{13}} , \tag{103}$$

$$F'_{24} = \frac{F_{24}}{\ell_{24}} . \tag{104}$$

5. The actual 6x6 partitions are

$$[K^d_{ij}] = \left[\begin{array}{c|c} k^d_{ij} & 0 \\ \hline 0 & 0 \end{array}\right] , \tag{105}$$

and

$$[K^d_{ji}] = [K^d_{ij}]^T . \tag{106}$$

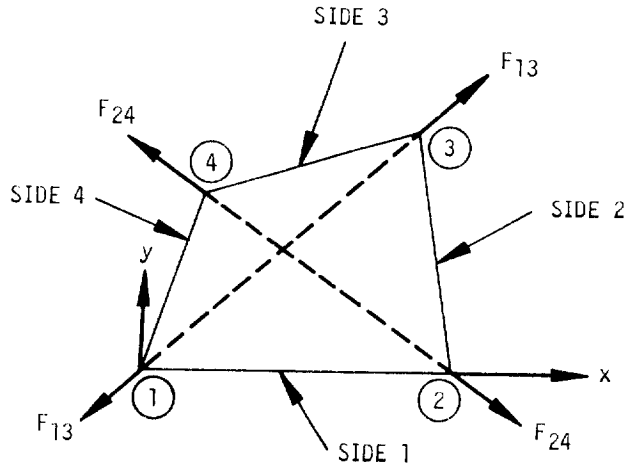


Figure 2. Shear panel element coordinate system and element forces.

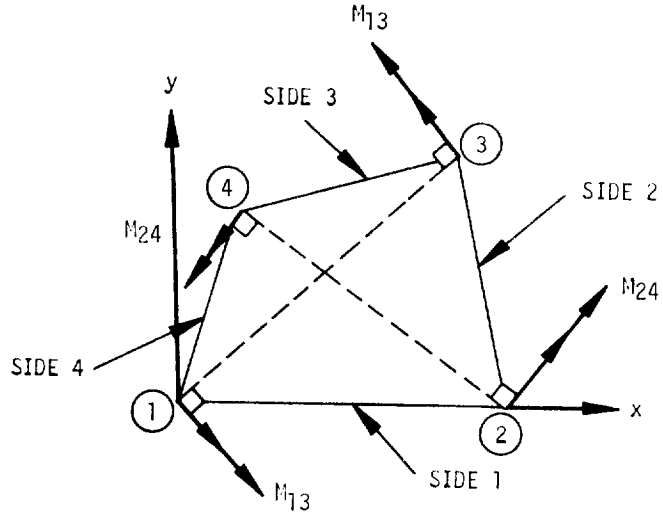


Figure 3. Twist panel element coordinate system and element forces.

4.87.4 TRMEM and QDMEM Elements

4.87.4.1 Input Data for the TRMEM and QDMEM Elements

1. ECPT entries for the TRMEM and QDMEM are:

Symbol		Description
TRMEM	QDMEM	
SIL_1	SIL_1	Scalar indices of the connected grid points.
SIL_2	SIL_2	
SIL_3	SIL_3	
	SIL_4	
N_i	N_i	Local coordinate system numbers and location coordinates in the basic system for the connected grid points.
X_i $i = 1,3$	X_i $i = 1,4$	
Y_i	Y_i	
Z_i	Z_i	
θ		Anisotropic material orientation angle
Mat I.D.		Material identification number
t		Thickness
μ		Nonstructural mass per unit area
t_μ		Temperature for material properties

2. Coordinate system data

The numbers N_i, X_i, Y_i and Z_i are used to calculate the 3 by 3 global-to-basic coordinate transformation matrices $[T_i]$ for points i = 1, 2, 3, and 4.

3. Material data

Symbol	Description
$[G_e]$	3x3 stress-strain matrix
ρ	Mass density
$\alpha_x, \alpha_y, \alpha_{xy}$	Three thermal expansion coefficients
T_o	Reference temperature
g_e	Structural damping coefficient
$\sigma_t, \sigma_c, \sigma_s$	Stress limits for tension, compression and shear

4.87.4.2 Basic Equations For TRMEM

1. The element coordinate system is defined by the following equations (see Figure 4)

$$\{V_{12}\} = \begin{Bmatrix} X_2 - X_1 \\ Y_2 - Y_1 \\ Z_2 - Z_1 \end{Bmatrix}, \tag{1}$$

$$\{V_{13}\} = \begin{Bmatrix} X_3 - X_1 \\ Y_3 - Y_1 \\ Z_3 - Z_1 \end{Bmatrix}, \tag{2}$$

$$\{i\} = \frac{\{V_{12}\}}{|\{V_{12}\}|}, \tag{3}$$

$$\{k\} = \frac{\{i\} \times \{V_{13}\}}{|\{i\} \times \{V_{13}\}|}, \tag{4}$$

$$\{j\} = \{k\} \times \{i\}. \tag{5}$$

2. The displacement transformation matrix from basic coordinates to in-plane coordinates is :

$$[E]^T = \begin{bmatrix} i_1 & i_2 & i_3 \\ j_1 & j_2 & j_3 \end{bmatrix}, \tag{6}$$

3. The coordinates of the points in the element coordinate system are:

$$x_1 = y_1 = y_2 = 0, \tag{7}$$

$$x_2 = |\{V_{12}\}|, \tag{8}$$

$$x_3 = \{V_{13}\}^T \{i\}, \tag{9}$$

$$y_3 = |\{i\} \times \{V_{13}\}|. \tag{10}$$

The area is:

$$A = \frac{1}{2} x_2 y_3. \tag{11}$$

4. The transformations from displacements at the points to strains are:

$$[C_1] = \begin{bmatrix} -\frac{1}{x_2} & 0 \\ 0 & \frac{1}{y_3}\left(\frac{x_3}{x_2} - 1\right) \\ \frac{1}{y_3}\left(\frac{x_3}{x_2} - 1\right) & -\frac{1}{x_2} \end{bmatrix}, \tag{12}$$

$$[C_2] = \begin{bmatrix} \frac{1}{x_2} & 0 \\ 0 & -\frac{x_3}{x_2 y_3} \\ -\frac{x_3}{x_2 y_3} & \frac{1}{x_2} \end{bmatrix}, \tag{13}$$

$$[C_3] = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{y_3} \\ \frac{1}{y_3} & 0 \end{bmatrix}. \tag{14}$$

4.87.4.3 Stiffness Matrix Calculation for TRMEM (Subroutine KTRMEM of Module SMA1)

1. The equation used in the stiffness matrix generation in global coordinates is:

$$[k_{ij}] = At([C_i][E]^T[T_i])^T[G_e]([C_j][E]^T[T_j]), \tag{15}$$

where "i" is the pivot point number, and j = 1, 2, 3 are the three connected points. $[k_{ij}]$ is a 3x3 matrix.

2. For use in the overall structural matrix, the matrices are expanded to 6x6 to form:

$$[K_{ij}] = \left[\begin{array}{c|c} k_{ij} & 0 \\ \hline 0 & 0 \end{array}\right]. \tag{16}$$

4.87.4.4 Mass Matrix Calculation for the TRMEM Element (Subroutine MASSTQ of Module SMA2)

The mass is generated by the following algorithm.

The vectors defining the sides are :

$$\{V_{12}\} = \begin{Bmatrix} X_2 - X_1 \\ Y_2 - Y_1 \\ Z_2 - Z_1 \end{Bmatrix}, \tag{17}$$

$$\{V_{13}\} = \begin{Bmatrix} X_3 - X_1 \\ Y_3 - Y_1 \\ Z_3 - Z_1 \end{Bmatrix}. \tag{18}$$

The area is :

$$A = \frac{1}{2} |\{V_{12}\} \times \{V_{13}\}| . \tag{19}$$

The mass at each point is :

$$m = \frac{A}{3} (\sigma t + \mu), \tag{20}$$

which is one-third of the total mass.

For each point the diagonal mass matrix is :

$$[M_{ii}] = \begin{bmatrix} m & & & 0 & & \\ & m & & & 0 & \\ & & m & & & 0 \\ 0 & & & 0 & & \\ & 0 & & & 0 & \\ & & 0 & & & 0 \end{bmatrix}, \quad i = 1,2,3. \tag{21}$$

4.87.4.5 Element Load Calculations For The TRMEM Element (Subroutine TRIMEM of Module SSG1)

Using the loading temperature on the three grid points t_1, t_2, and t_3 given in the GPTT data block, the triangular membrane routine generates force vectors by the equation:

$$\{P_i\} = At[T_i]^T[E]\,[C_i]^T[G_e]\{\alpha\}\bar{t}\ , \qquad \text{for } i = 1, 2, 3 \tag{22}$$

where

$$\bar{t} = \frac{1}{3}(t_1 + t_2 + t_3) - T_o\ , \tag{23}$$

and $\{P_i\}$ is a 3x1 vector.

The forces are placed in the PG load vector data block.

4.87.4.6 Element Stress Calculations For The TRMEM Element (Subroutines STRME1 and STQME2 of Module SDR2)

1. Calculations performed in STRME1 (Phase 1 calculations).

   a. Using the formulae given in section 4.87.4.2, calculate the following terms:

$[C_i]$	i = a, b, c	(3x2)
$[T_i]$	i = a, b, c	(3x3)
$[E]$		(3x2)
$[G_e]$		(3x3)

The transformations from displacements to stress are:

$$[S_i] = [G_e][C_i][E]^T[T_i]\cdot \tag{24}$$

The temperature to stress relation is:

$$\{S_t\} = -[G_e]\{\alpha\} \tag{25}$$

where

$$\{\alpha\} = \alpha \begin{Bmatrix} 1 \\ 1 \\ 0 \end{Bmatrix}, \tag{26}$$

for isotropic materials. $\{\alpha\}$ is input by the user for anisotropic materials.

2. Calculations performed by STQME2 (Phase 2 calculations)

The equation for stress is:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \left[\sum_{i=a,b,c} [S_i]\{u_{gi}\} \right] + \{S_t\} \left[\frac{t_a + t_b + t_c}{3} - t_o \right], \tag{27}$$

where t_a, t_b and t_c are the loading temperatures obtained from the GPTT data block.

The principal stresses are:

$$\sigma_1 = \left(\frac{\sigma_x + \sigma_y}{2} \right) + \sqrt{\left(\frac{\sigma_x - \sigma_y}{2} \right)^2 + \sigma_{xy}^2}, \tag{28}$$

$$\sigma_2 = \left(\frac{\sigma_x + \sigma_y}{2} \right) - \sqrt{\left(\frac{\sigma_x - \sigma_y}{2} \right)^2 + \sigma_{xy}^2}, \tag{29}$$

$$\theta = \frac{1}{2} \arctan \left(\frac{2\sigma_{xy}}{\sigma_x - \sigma_y} \right) \text{ (in degrees)}, \tag{30}$$

where θ is limited to: $-90^o \leq \theta \leq 90^o$

The maximum shear is :

$$\tau = \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \sigma_{xy}^2} \qquad (31)$$

4.87.4.7 Differential Stiffness Matrix Calculations for the TRMEM Element (Subroutine DTRMEM of Module DSMG1)

1. Input Data.

   ECPT for element

   i - Pivot point scalar index

   $\{u_1\}$, $\{u_2\}$, $\{u_3\}$ - Displacements of pivots on triangle (UGV)

   t_i - Loading temperature at the grid points of the element (GPTT)

   CSTM - coordinate systems

   MPT - Material properties table

2. Output Data.

   $[K^d_{i1}]$, $[K^d_{i2}]$, $[K^d_{i3}]$ - partitions of the differential stiffness matrix.

3. Solution Algorithm.

   a. The planar stresses in the element, σ_x, σ_y and σ_{xy}, are calculated as in the SDR2 (Stress Data Recovery) module. The following data are saved for use in the differential stiffness calculation :

   $\{i\}$, $\{j\}$, $\{k\}$ - Unit vectors defining the element coordinate system.

   A, t - Area and thickness

   x_2, x_3, y_3 - Locations of the points, element coordinates

   $[T_1]$, $[T_2]$, $[T_3]$ - Global-to-basic coordinate transformations

   σ_x, σ_y, σ_{xy} - Stresses in element system

b. In generalized coordinates the differential stiffness matrix is :

$$[K_\omega^d] = At \begin{bmatrix} \sigma_y & -\sigma_{xy} & 0 \\ -\sigma_{xy} & \sigma_x & 0 \\ 0 & 0 & \sigma_x + \sigma_y \end{bmatrix} . \quad (32)$$

c. The transformation from displacements at the points to rigid body rotations are :

$$[C_1^d] = \begin{bmatrix} 0 & 0 & \gamma_3 - \gamma_2 \\ 0 & 0 & \gamma_1 \\ \frac{\gamma_2 - \gamma_3}{2} & -\frac{\gamma_1}{2} & 0 \end{bmatrix} , \quad (33)$$

$$[C_2^d] = \begin{bmatrix} 0 & 0 & -\gamma_3 \\ 0 & 0 & -\gamma_1 \\ \frac{\gamma_3}{2} & \frac{\gamma_1}{2} & 0 \end{bmatrix} , \quad (34)$$

$$[C_3^d] = \begin{bmatrix} 0 & 0 & \gamma_2 \\ 0 & 0 & 0 \\ -\frac{\gamma_2}{2} & 0 & 0 \end{bmatrix} , \quad (35)$$

where

$$\gamma_1 = \frac{1}{x_2} , \tag{36}$$

$$\gamma_2 = \frac{1}{y_2} , \tag{37}$$

$$\gamma_3 = \frac{x_3}{x_2 y_3} . \tag{38}$$

d. The partitions (3x3) of the differential stiffness matrix in global coordinates are :

$$[K_{ij}] = ([C_i^d] [E^d]^T [T_i])^T [K_\omega^d] [C_j^d] [E^d]^T [T_j] , \tag{39}$$

i = pivot point

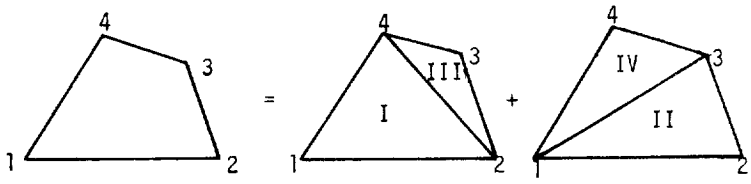
j = 1, 2 and 3

where

$$[E^d]^T = \begin{bmatrix} \{i\}^T \\ \{j\}^T \\ \{k\}^T \end{bmatrix} \quad (3x3) . \tag{40}$$

4.87.4.8 General Calculations for the QDMEM by the QDMEM Driver Routines (Subroutines KQDMEM of Module SMA1, SQDMEl of Module SDR2, DQDMEM of Module DSMG1).

1. The quadrilateral is divided into four triangles as shown in the figure below:



The thickness used for each triangle is one-half that given for the quadrilateral. Since no special calculation time is saved by generating a unique element coordinate system, the basic locations of the points are used to calculate individual coordinate systems for the triangles.

An integer mapping matrix [M] containing the quadrilateral point numbers is used to convert point numbers for the triangles to point numbers for the quadrilateral.

$$[M] = \begin{array}{c} \begin{matrix} \text{Triangle Point No.} & & & \text{Triangle No.} \\ \underline{a} & \underline{b} & \underline{c} & \end{matrix} \\ \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 1 \\ 3 & 4 & 2 \\ 4 & 1 & 3 \end{bmatrix} \begin{matrix} (I) \\ (II) \\ (III) \\ (IV) \end{matrix} \end{array} \qquad (41)$$

The data corresponding to the point numbers in each row of the matrix are transferred to the triangular membrane routine. The pivot grid point i is also transferred.

2. Material orientation for subtriangles.

The material orientation angle for the QDMEM element must be transformed to a set of angles related to the base of each subtriangle. This requires the following steps:

1. The element coordinate system is defined as follows:

$$\{V_i\} = \begin{Bmatrix} X_i \\ Y_i \\ Z_i \end{Bmatrix} \quad i = 1, 2, 3, 4, \tag{42}$$

$$\{d_{21}\} = \{V_2\} - \{V_1\} \quad , \tag{43}$$

$$\{i\} = \frac{\{d_{21}\}}{|\{d_{21}\}|} \quad , \tag{44}$$

$$\{d_{41}\} = \{V_4\} - \{V_1\} \quad , \tag{45}$$

$$\{k\} = \frac{\{i\} \times \{d_{41}\}}{|\{i\} \times \{d_{41}\}|} \quad , \tag{46}$$

$$\{j\} = \{k\} \times \{i\} \quad . \tag{47}$$

The material is oriented for each triangle as follows:

$$s_1 = \sin(\theta) \, , \tag{48}$$

$$c_1 = \cos(\theta) \, , \tag{49}$$

$$\{p\} = c_1 \{i\} + s_1 \{j\} \, , \tag{50}$$

$$\{V_{II}^t\} = \{V_3\} - \{V_2\}, \tag{51}$$

$$\{V_{III}^t\} = \{V_4\} - \{V_3\} \, , \tag{52}$$

$$\{V_{IV}^t\} = \{V_1\} - \{V_4\} \quad , \tag{53}$$

$$c_i = \frac{\{V_i^t\}^T\{p\}}{|\{V_i^t\}|} = \cos(\theta_i), \qquad (54)$$

$$i = II, III \text{ and } IV$$

$$s_i = \frac{(\{V_i^t\} \times \{p\})^T\{k\}}{|\{V_i^t\}|} = \sin(\theta_i). \qquad (55)$$

The values s_i and c_i may be passed to the triangular membrane subroutines in lieu of the angles θ_i.

4.87.4.9 Stiffness Matrix Calculations for the QDMEM.

Three stiffness matrices, which the triangular membrane routine calculates for each sub-triangle, are added to the four matrices which will be output. (Note: only three triangles are needed for each pivot point.) For example, consider the case where point 2 is the pivot grid point. (i.e., the second SIL value in the grid point connection list equals the pivot grid point SIL value). Triangle I is calculated by entering the geometry and property data for the 1, 2 and 4 points on the quadrilateral, with number 2 as the pivot point. The outputs from the stiffness matrix generation routines for the TRMEM are:

$$[K_{21}],\ [K_{22}],\ [K_{24}]$$

Data for triangles II and III are also entered, and their corresponding matrix partitions are added. Triangle number IV is not connected to point 2.

4.87.4.10 Element Stress Calculations for the QDMEM (Subroutine SQDME1 and STQME2 of Module SDR2).

The solution for stress in the quadrilateral involves two phases. In the first phase (SQDME1) the triangular membrane partitions are solved for their stress-displacement matrices. These matrices are modified to correspond to the element coordinate system. They

are added together to form four 3x3 matrices relating displacements in global coordinates to element stress. A vector is also calculated which transforms temperature to stress.

The second phase (STQME2) involves the acquisition of the displacement and temperature data and the calculation of the net stress.

The following steps are used to set up an element coordinate system and obtain triangle to element stress transformations.

Phase 1

1. The following quantities are calculated:

$$\{V_i\} = \begin{Bmatrix} X_i \\ Y_i \\ Z_i \end{Bmatrix} \quad i = 1, 2, 3, 4 , \tag{56}$$

$$\{d_1\} = \{V_3\} - \{V_1\} , \tag{57}$$

$$\{d_2\} = \{V_4\} - \{V_2\} , \tag{58}$$

$$\{k\} = \frac{\{d_1\} \times \{d_2\}}{|\{d_1\} \times \{d_2\}|} , \tag{59}$$

$$\{a_{12}\} = \{V_2\} - \{V_4\} , \tag{60}$$

$$\{a_{23}\} = \{V_3\} - \{V_2\} , \tag{61}$$

$$\{a_{41}\} = \{V_1\} - \{V_4\} , \tag{62}$$

$$\{a_{34}\} = \{V_4\} - \{V_3\} , \tag{63}$$

$$h = \{a_{12}\}^T \{k\} \quad . \tag{64}$$

h is the perpendicular distance between the diagonals. The mean plane of the element lies halfway between the diagonals.

2. The unit vectors along the edges of the four triangles, projected on the mean plane, are calculated from:

$$x_2 = |\{a_{12}\} - h\{k\}| \quad , \tag{65}$$

$$\{i\} = \frac{\{a_{12}\} - h\{k\}}{x_2} \quad , \tag{66}$$

$$\{j\} = \{k\} \times \{j\} \quad , \tag{67}$$

$$[R] = \begin{bmatrix} \{i\}^T \\ \{j\}^T \end{bmatrix} \quad (2x3) \quad , \tag{68}$$

$$\{v_{12}\} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} , \tag{69}$$

$$\{v_{ij}\} = [R]\{a_{ij}\} \quad ij = 23, 34, 41 \quad , \tag{70}$$

$$\{w^I\} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} , \tag{71}$$

$$\{w^{II}\} = \frac{1}{|\{v_{23}\}|} \{v_{23}\} \quad , \tag{72}$$

$$\{w^{III}\} = \frac{1}{|\{v_{34}\}|} \{v_{34}\} \quad , \tag{73}$$

$$\{w^{IV}\} = \frac{1}{|\{v_{41}\}|} \{v_{41}\} \quad , \tag{74}$$

3. For each triangular membrane, β = I, II, III, IV, of the quadrilateral, the subroutine STRME1 is called to calculate the three stress functions $[S_i]$ where

$$\left\{ \begin{matrix} \sigma_1 \\ \sigma_2 \\ \sigma_{12} \end{matrix} \right\}_{\text{triangle}} = \sum_{i=1}^{3} [S_i] \{u_{gdi}\} , \tag{75}$$

where $\{u_{gdi}\}$ are the displacements in global coordinates of the points on the triangle. $[S_i^\beta]$ is calculated using the full thickness of the panel for triangle β.

4. The stress functions, Equation 24, are transformed to the element coordinates by the matrix $[T^\beta]$:

$$[S_i^\beta]_{\text{element}} = [T^\beta][S_i^\beta], \tag{76}$$

where

$$[T^\beta] = \begin{bmatrix} w_1^2 & w_2^2 & -2w_1w_2 \\ w_2^2 & w_1^2 & 2w_1w_2 \\ w_1w_2 & -w_1w_2 & w_2^2-w_1^2 \end{bmatrix} , \tag{77}$$

and

$$\left\{ \begin{matrix} w_1 \\ w_2 \end{matrix} \right\} = \{w^\beta\} \tag{78}$$

for triangle β = I, II, III, IV.

5. Using the mapping matrix [M], Equation 41, the matrices are added. The actual equations are:

$$[\bar{S}_1] = \tfrac{1}{4}([S_a^I] + [S_c^{II}] + [S_b^{IV}]) , \tag{79}$$

$$[\bar{S}_2] = \tfrac{1}{4}([S_b^I] + [S_a^{II}] + [S_c^{III}]) , \tag{80}$$

$$[\bar{S}_3] = \frac{1}{4}([S_b^{II}] + [S_a^{III}] + [S_c^{IV}]) , \tag{81}$$

$$[\bar{S}_4] = \frac{1}{4}([S_c^{I}] + [S_b^{III}] + [S_a^{IV}]) , \tag{82}$$

$$\{\bar{S}_t\} = \frac{1}{4}(\{S_t^{I}\} + \{S_t^{II}\} + \{S_t^{III}\} + \{S_t^{IV}\}). \tag{83}$$

where $[S_i^\beta]$ i = a, b, c, are the stress matrices in Equation 76 and $\{S_t^\beta\}$ is the vector in Equation 25.

Phase 2

1. In phase 2 the stresses are calculated from:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \{\bar{S}_t\}\bar{t} + \sum_{i=1}^{4} [\bar{S}_i] \{u_{gdi}\} , \tag{84}$$

where

$$\bar{t} = \frac{1}{4} \sum_{i=1}^{4} (t_i - T_o) , \tag{85}$$

and t_i are the loading temperatures at the grid points, obtained from the GPTT data block.

2. The principal stresses and the angles are calculated in exactly the same manner as for the TRMEM element.

4.87.4.11 Mass Matrix Generation for the QDMEM Element (Subroutine MASSTQ of Module SMA2).

The mass is generated by the following algorithm.

The vectors defining the sides and the diagonals are:

$$\{V_{ij}\} = \begin{Bmatrix} X_j - X_i \\ Y_j - Y_i \\ Z_j - Z_i \end{Bmatrix} , \quad ij = 12, 23, 34, 41, 13, 24 \tag{86}$$

The area of the subtriangles defined by the integer mapping matrix [M], Equation 41, is:

$$A_I = \frac{1}{2} \left| \{V_{14}\} \times \{V_{12}\} \right| , \tag{87}$$

$$A_{II} = \frac{1}{2} \left| \{V_{12}\} \times \{V_{23}\} \right| , \tag{88}$$

$$A_{III} = \frac{1}{2} \left| \{V_{23}\} \times \{V_{34}\} \right| , \tag{89}$$

$$A_{IV} = \frac{1}{2} \left| \{V_{34}\} \times \{V_{14}\} \right| . \tag{90}$$

The area of the quadrilateral is:

$$A_q = \frac{1}{2} \left| \{V_{13}\} \times \{V_{24}\} \right| . \tag{91}$$

The mass at each point is:

$$m_1 = \frac{(A_I + A_q)}{3} (\mu + pt) , \tag{92}$$

$$m_2 = \frac{(A_{II} + A_q)}{3} (\mu + pt) , \tag{93}$$

$$m_3 = \frac{(A_{III} + A_q)}{3} (\mu + pt) , \tag{94}$$

$$m_4 = \frac{(A_{IV} + A_q)}{3} (\mu + pt) . \tag{95}$$

For each point, the diagonal mass matrix is:

$$[M_{ii}] = \begin{bmatrix} m_i & & & \\ & m_i & & 0 \\ & & m_i & \\ & 0 & & 0 \end{bmatrix} , \quad i = 1, 2, 3, 4 . \tag{96}$$

4.87.4.12 Thermal Load Computation For The QDMEM.

The thermal loads are calculated using the triangular membrane routine. The EST data are rearranged to correspond to each of the four subtriangles, and each triangle produces a load in global coordinates.

4.87.4.13 Differential Stiffness Computations For The QDMEM (Subroutines DQDMEM and DTRMEM of Module DSMG1).

The differential stiffness matrices for the QDMEM element are generated by rearranging the ECPT data into four sets of TRMEM data. The TRMEM differential stiffness routine calculates the stresses, generates the differential stiffness matrix partitions in global coordinates and inserts them in the overall matrix.

4.87.4.14 Piecewise Linear Analysis Calculations (Subroutines PSTRM and PSQDM of Module PLA3 and Subroutines PKTRM and PKQDM of Module PLA4)

The additional ECPTNL and ESTNL entries are:

ε_o^* - The previously computed strain value once removed.

ε^* - The previously computed strain value.

E^* - The previously computed modulus of elasticity.

σ_x^*, σ_y^*, σ_{xy}^* } The previously computed membrane stresses

All of the above values are initially zero with the exception of E^*, which is initially the original modulus of elasticity present on a MAT1 card.

For both PLA3 and PLA4, the element stress matrix calculations are generated in the same manner as Equation 24 of section 4.87.4.6 (Equations 79 through 82 of section 4.87.4.8 for the QDMEM), with the exception that for all DMAP loops (of the Piecewise Linear Analysis Rigid Format) after the first, the 3 by 3 material properties matrix $[G_e]$ is replaced by a stress-dependent 3 by 3 material properties matrix $[G_p]$ defined as follows

$$[G_p] = E_o \begin{bmatrix} 1+s_x^2F & -\nu+s_xs_yF & 2\sigma_{xy}^*s_xF \\ & 1+s_y^2F & 2\sigma_{xy}^*s_yF \\ (\text{sym}) & & 2(1+\nu)+4F\sigma_{xy}^{*2} \end{bmatrix}^{-1} , \qquad (97)$$

where

$$\tau_o = \left[\sigma_x^{*2} - \sigma_x^*\sigma_y^* + \sigma_y^{*2} + 3\sigma_{xy}^{*2}\right]^{1/2} , \qquad (98)$$

$$F = \frac{9(E_o - E^*)}{4\tau_o^2E^*} , \qquad (99)$$

$$s_x = \frac{2\sigma_x^* - \sigma_y^*}{3} \quad , \tag{100}$$

$$s_y = \frac{2\sigma_y^* - \sigma_x^*}{3} \quad , \tag{101}$$

and E_o and ν are the linear type 1 material properties. If $E^* = 0$, or $\tau_o = 0$, then $[G_p] = [0]$.

Calculate the incremental element stresses:

$$\begin{Bmatrix} \Delta\sigma_x \\ \Delta\sigma_y \\ \Delta\sigma_{xy} \end{Bmatrix} = \left[\sum_{i=a,b,c} [S_i]\{\Delta u_{gi}\} \right] \quad , \tag{102}$$

where $[S_i]$ is given in Equation 24 of section 4.87.4.6, and $\{\Delta u_{gi}\}$ are the 3 by 1 translational displacement vectors.

Define the element stresses for output and for updating the ECPTNL and ESTNL:

$$\sigma_{x1} = \sigma_x^* + \Delta\sigma_x \quad , \tag{103}$$

$$\sigma_{y1} = \sigma_y^* + \Delta\sigma_y \quad , \tag{104}$$

$$\sigma_{xy1} = \sigma_{xy}^* + \Delta\sigma_{xy} \quad . \tag{105}$$

In PLA3, using the element stresses above, the principal stresses are calculated in the same manner as in Equations 28 through 31 in section 4.87.4.6.

Estimate the next elastic coefficients as defined by the following equations:

$$\tau_1 = \left[\sigma_{x1}^2 - \sigma_{x1}\sigma_{y1} + \sigma_{y1}^2 + 3\sigma_{xy1}^2 \right]^{1/2} \quad , \tag{106}$$

$$\varepsilon_1 = f^{-1}(\tau_1) \quad , \tag{107}$$

where f is the tabular stress-strain function. (When τ_1 is outside the range of the function, define $E_1 = 0$, $\varepsilon_1 = \varepsilon^*$, and $\varepsilon^* = \varepsilon_0^*$).

Calculate:

$$\Delta\varepsilon = \varepsilon_1 - \varepsilon^* \quad , \tag{108}$$

$$\Delta\varepsilon^* = \varepsilon^* - \varepsilon_0^* \quad , \tag{109}$$

$$\varepsilon_2 = \varepsilon_1 + \gamma(\Delta\varepsilon) \quad , \tag{110}$$

where γ is a load ratio parameter calculated by the module driver (PLA3 or PLA4).

Calculate:

$$\tau_2 = f(\varepsilon_2) \quad , \tag{111}$$

where f is the tabular stress-strain function.

Then the estimated next modulus of elasticity, E_1, is given by:

$$E_1 = \begin{cases} \dfrac{\tau_2 - \tau_1}{\varepsilon_2 - \varepsilon_1} \text{ , for } \varepsilon_2 \neq \varepsilon_1 \\ 0 \text{ , for } \varepsilon_2 = \varepsilon_1 . \end{cases} \tag{112}$$

The new ESTNL and ECPTNL entries are:

$$\varepsilon_0^* = \varepsilon^* \quad , \tag{113}$$

$$\varepsilon^* = \varepsilon_1 \quad , \tag{114}$$

$$E^* = E_1 \quad , \tag{115}$$

$$\sigma_x^* = \sigma_{x1} \quad , \tag{116}$$

$$\sigma_y^* = \sigma_{y1} \quad , \tag{117}$$

$$\sigma_{xy}^* = \sigma_{xy1} \quad . \tag{118}$$

In module PLA4, the element stiffness matrices are calculated in the same manner as Equations 15 and 16 of section 4.87.4.3 (or as in section 4.87.4.9 for the QDMEM), with the

exception that $[G_e]$ matrix is replaced by the $[G_p]$ matrix (Equation 97). In the calculation for $[G_p]$, E_1 (Equation 112) is used for E^*, and the newly calculated membrane stresses (Equations 103, 104 and 105) are used in place of σ_x^*, σ_y^* and σ_{xy}^*.

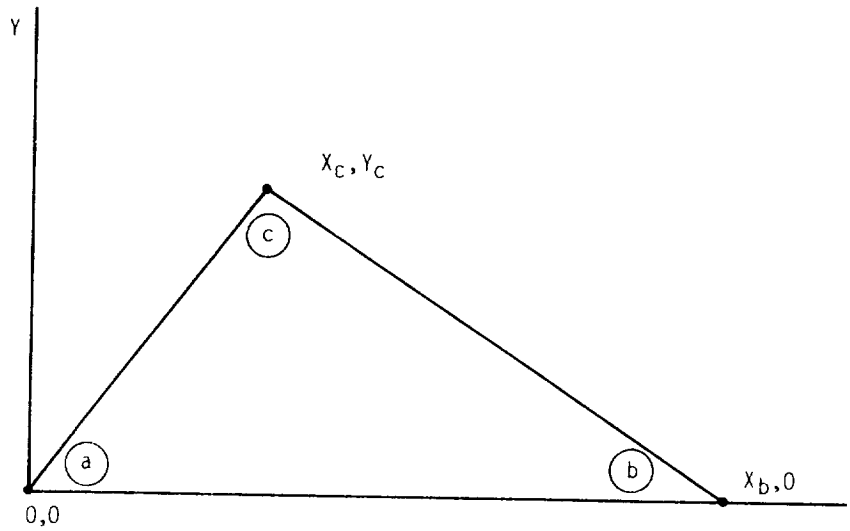


Figure 4. Triangular membrane element.

4.87.5 The TRBSC, TRPLT and QDPLT Elements

4.87.5.1 Input Data for the TRBSC and TRPLT Elements.

1. The ECPT/EST entries for the TRBSC and TRPLT are:

Symbol	Description
SIL_1, SIL_2, SIL_3	Scalar indices for the connected grid points
N_i, X_i, Y_i, Z_i $i = 1, 2, 3$	Reference numbers for local coordinate system and locations in basic coordinates of the three connected grid points
I	Bending moment of inertia per unit width
t	Effective thickness for transverse shear
Mat $Id._b$	Material property identification number for bending
Mat. $Id._s$	Material property identification number for shear
θ	Material orientation angle
μ	Nonstructural mass per area
Z_1, Z_2	Fiber distances for stress calculations
t_μ	Temperature of element for material properties

2. ECPT entries for the QDPLT.

The entries are the same as those for the TRPLT except that four points are used.

3. Coordinate system data

Using N_i, X_i, Y_i, Z_i and the CSTM data block the 3 by 3 global-to-basic coordinate transformation matrices $[T_i]$ are produced for each point i via subroutines TRANSD or TRANSS.

4. Material data

Using the material property identification numbers, the orientation angle, the element temperature and the MPT and DIT data blocks. the following data are calculated:

Symbol		Description
for Mat Id_b	$[G_b]$	3x3 elastic property matrix
	g_e	Structural damping coefficient
for Mat Id_s	G_s	Shear coefficient

For the TRPLT and the QDPLT, the orientation of the material relative to each sub-triangle must be calculated from the geometry and the orientation given for the whole element. The details of this calculation are given in the next section.

4.87.5.2 General Calculation for the TRBSC Element

1. The coordinate system is defined by the three connected points a, b and c. {i}, {j} and {k} are the unit vectors along the x, y and z axis in basic coordinates. X_i, Y_i and Z_i are the location coordinates of the three points, i=a, b, c. (The element coordinate system for the basic bending triangle is shown in Figure 2 of section 5.8 of the Theoretical Manual).

$$\{V_{ab}\} = \begin{Bmatrix} X_b - X_a \\ Y_b - Y_a \\ Z_b - Z_a \end{Bmatrix}, \tag{1}$$

$$\{V_{ac}\} = \begin{Bmatrix} X_c - X_a \\ Y_c - X_a \\ Z_c - Z_a \end{Bmatrix}, \tag{2}$$

The x axis is defined by the unit vector:

$$\{i\} = \frac{\{V_{ab}\}}{|\{V_{ab}\}|} . \tag{3}$$

Calculate:

$$\{k\} = \frac{\{i\} \times \{V_{ac}\}}{|\{i\} \times \{V_{ac}\}|} . \tag{4}$$

The y axis is defined by the unit vector,

$$\{j\} = \{k\} \times \{i\} . \tag{5}$$

2. The locations of the points in element coordinates are:

$$x_a = y_a = y_b = 0 , \tag{6}$$

$$x_b = |\{V_{ab}\}| , \tag{7}$$

$$x_c = \{V_{ac}\}^T \{i\} , \tag{8}$$

$$y_c = \{V_{ac}\}^T \{j\} = |\{i\} \times \{V_{ac}\}| . \tag{9}$$

3. The 3 x 6 transformation matrix from the six displacements in element coordinates to the three degrees of freedom used in the plate is:

$$[E]^T = \begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & i_1 & i_2 & i_3 \\ 0 & 0 & 0 & j_1 & j_2 & j_3 \end{bmatrix} . \tag{10}$$

4. The coefficients used for the plate are:

$$[D] = I[G_b] , \tag{11}$$

$$[G_2] = t_s \begin{bmatrix} G_s & 0 \\ 0 & G_s \end{bmatrix} . \tag{12}$$

where $[G_b]$ is the 3 x 3 material matrix for bending and G_s is the coefficient for transverse shear. The area of the triangle, A, the locations of the c.g., $\bar{x}$ and $\bar{y}$, and the radii of gyration about the origin, ρ_x^2, ρ_y^2 and ρ_{xy}^2 are given by:

$$A = \frac{1}{2} x_b y_c \quad , \tag{13}$$

$$\bar{x} = \frac{1}{3} (x_c + x_b) \, , \tag{14}$$

$$\bar{y} = \frac{1}{3} y_c \quad , \tag{15}$$

$$\rho_x^2 = \frac{1}{6} (x_b^2 + x_b x_c + x_c^2) \quad , \tag{16}$$

$$\rho_y^2 = \frac{1}{6} y_c^2 \quad , \tag{17}$$

$$\rho_{xy}^2 = \frac{y_c}{12} (x_b + 2x_c) \, . \tag{18}$$

5. The stiffness matrix in generalized coordinates $\{q\}$ is:

$$[K^{\wedge}] = 4A \begin{bmatrix} D_{11} & D_{13} & D_{12} & 3\bar{x}D_{11} & \bar{x}D_{12}+2\bar{y}D_{13} & 3\bar{y}D_{12} \\ & D_{33} & D_{23} & 3\bar{x}D_{13} & \bar{x}D_{23}+2\bar{y}D_{33} & 3\bar{y}D_{23} \\ & & D_{22} & 3\bar{x}D_{12} & \bar{x}D_{22}+2\bar{y}D_{23} & 3\bar{y}D_{22} \\ & & & 9\rho_x^2 D_{11} & 3\rho_x^2 D_{12}+6\rho_{xy}^2 D_{13} & 9\rho_{xy}^2 D_{12} \\ & & & & \rho_x^2 D_{22}+4\rho_{xy}^2 D_{23}+4\rho_y^2 D_{33} & 3\rho_{xy}^2 D_{22}+6\rho_y^2 D_{23} \\ & & & & & 9\rho_y^2 D_{22} \end{bmatrix} \tag{19}$$

SYMMETRICAL

6. The transformation from generalized coordinates to grid point displacements (relative to point "a" of the triangle) with no transverse shear is:

$$[\bar{H}] = \begin{bmatrix} x_b^2 & 0 & 0 & x_b^3 & 0 & 0 \\ 0 & x_b & 0 & 0 & 0 & 0 \\ -2x_b & 0 & 0 & -3x_b^2 & 0 & 0 \\ x_c^2 & x_c y_c & y_c^2 & x_c^3 & x_c y_c^2 & y_c^3 \\ 0 & x_c & 2y_c & 0 & 2x_c y_c & 3y_c^2 \\ -2x_c & -y_c & 0 & -3x_c^2 & -y_c^2 & 0 \end{bmatrix}. \quad (20)$$

where $\{q\} = [\bar{H}]\{u\}$ (no transverse shear)

7. If transverse shear effects are to be calculated ($G_s\ t_s \neq 0$), the following steps are followed:

a. Define

$$[J] = [G_2]^{-1}. \quad (21)$$

b. Calculate the transformation matrix of the shear coordinates:

$$[H_{\gamma q}] = -\left[\begin{array}{ccc|ccc|c} 0 & 0 & 0 & 6(J_{11}D_{11}+J_{12}D_{13}) & J_{11}(2D_{12}+4D_{33})+6J_{12}D_{23} & 6(J_{11}D_{23}+J_{12}D_{22}) \\ 0 & 0 & 0 & 6(J_{12}D_{11}+J_{22}D_{13}) & J_{12}(2D_{12}+4D_{33})+6J_{22}D_{23} & 6(J_{12}D_{23}+J_{22}D_{22}) \end{array}\right], \quad (22)$$

where

$$\begin{Bmatrix} \gamma_x \\ \gamma_y \end{Bmatrix} = [H_{\gamma q}] \{q\} . \tag{23}$$

c. The stiffness matrix of the shear terms is added to the bending stiffness matrix.

$$[K^q] = [K^X] + A[H_{\gamma q}]^T [G_2] [H_{\gamma q}] . \tag{24}$$

d. The effects of shear deflection on the transformation from general to displacement coordinates is calculated:

$$[H] = [\bar{H}] - \begin{bmatrix} 0 & 0 & 0 & x_b H_{\gamma q_{14}} & x_b H_{\gamma q_{15}} & x_b H_{\gamma q_{16}} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \begin{matrix} x_c H_{\gamma q_{14}} \\ +y_c H_{\gamma q_{24}} \end{matrix} & \begin{matrix} x_c H_{\gamma q_{15}} \\ +y_c H_{\gamma q_{25}} \end{matrix} & \begin{matrix} x_c H_{\gamma q_{16}} \\ +y_c H_{\gamma q_{26}} \end{matrix} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{25}$$

If no shear exists, $[H] = [\bar{H}]$

8. The matrix [H] is inverted:

$$[H_{qu}] = [H]^{-1} . \tag{26}$$

9. The rigid body effects are given by the matrices $[B_b]$ and $[B_c]$ defined as follows:

$$[B_b] = \begin{bmatrix} 1 & 0 & -x_b \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{27}$$

$$[B_c] = \begin{bmatrix} 1 & y_c & -x_c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{28}$$

10. The 3x3 stiffness matrix partitions in element coordinates are calculated as follows:

$$[K] = [H^{-1}]^T [K^q] [H^{-1}], \tag{29}$$

$$[K] \Longrightarrow \left[\begin{array}{c|c} k_{bb} & k_{bc} \\ \hline k_{cb} & k_{cc} \end{array}\right], \tag{30}$$

$$[k_{ca}] = -[k_{cb}] [B_b] - [k_{cc}] [B_c], \tag{31}$$

$$[k_{ba}] = -[k_{bb}] [B_b] - [k_{bc}] [B_c], \tag{32}$$

$$[k_{aa}] = -[B_b]^T [k_{ba}] - [B_c]^T [k_{ca}], \tag{33}$$

$$[k_{ac}] = [k_{ca}]^T, \tag{34}$$

$$[k_{ab}] = [k_{ba}]^T. \tag{35}$$

4.87.5.3 Stiffness Matrix Calculations for the TRBSC Element (Subroutine KTRBSC of Module SMA1).

1. If the basic triangle is used by itself as a TRBSC element, the stiffness matrices are:

$$[K_{ij}] = \left[\begin{array}{c|c} T_i^T & 0 \\ \hline 0 & T_i^T \end{array}\right] [E] [k_{ij}] [E]^T \left[\begin{array}{c|c} T_j & 0 \\ \hline 0 & T_j \end{array}\right], \tag{36}$$

where

i = a, b and c

j = a, b and c

2. The structural damping matrices are calculated using g_e, the structural damping coefficient, for the referenced bending material. The 6 by 6 damping matrix partitions are:

$$[K^4_{ij}] = g_e [K_{ij}] \quad . \tag{37}$$

4.87.5.4 Stress Calculations for the TRBSC Element.

The stress calculations involve two phases. The first phase is used to calculate the matrix relations between element forces and grid point displacements.

1. The relation between element forces and generalized coordinates is:

$$[k_s] = \begin{bmatrix} 2D_{11} & 2D_{13} & 2D_{12} & 6\bar{x}D_{11} & 2\bar{x}D_{12}+4\bar{y}D_{13} & 6\bar{y}D_{12} \\ 2D_{12} & 2D_{23} & 2D_{22} & 6\bar{x}D_{12} & 2\bar{x}D_{22}+4\bar{y}D_{23} & 6\bar{y}D_{22} \\ 2D_{13} & 2D_{33} & 2D_{23} & 6\bar{x}D_{13} & 2\bar{x}D_{23}+4\bar{y}D_{33} & 6\bar{y}D_{23} \\ 0 & 0 & 0 & -6D_{11} & -2D_{12}+4D_{33} & -6D_{23} \\ 0 & 0 & 0 & -6D_{13} & -6D_{23} & -6D_{22} \end{bmatrix} , \tag{38}$$

where

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \\ V_x \\ V_y \end{Bmatrix} = [K_s] \{q\} \quad . \tag{39}$$

Note: When the basic triangle routine is used for stress recovery in the TRPLT or QDPLT, the values $\bar{x}$ and $\bar{y}$ in the above matrix are replaced by x_c and y_c or x_q and y_q.

2. The matrix [H] is calculated and inverted. The [B] matrix is calculated ([B] is a 6x3 matrix, the $[B_b]$ matrix (Equation 27) comprising the first three rows and the $[B_c]$ matrix, Equation 28, comprising the last three rows). The [E] matrix and the global-to-basic transformation $[T_a]$, $[T_b]$, $[T_c]$ are generated. $[H]^{-1}$ is partitioned.

$$[H]^{-1} = [H_{1b} \mid H_{1c}]. \tag{40}$$

The element force - global displacement matrices are:

$$[S_a] = -[k_s][H]^{-1}[B][E]^T \left[\begin{array}{c|c} T_a & 0 \\ \hline 0 & T_a \end{array}\right], \tag{41}$$

$$[S_b] = [k_s][H_{1b}][E]^T \left[\begin{array}{c|c} T_b & 0 \\ \hline 0 & T_b \end{array}\right], \tag{42}$$

$$[S_c] = [k_s][H_{1c}][E]^T \left[\begin{array}{c|c} T_c & 0 \\ \hline 0 & T_c \end{array}\right]. \tag{43}$$

3. The second stage of stress calculations involves the use of the displacement vectors $\{u_a\}$, $\{u_b\}$ and $\{u_c\}$. The element forces are:

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \\ V_x \\ V_y \end{Bmatrix} = \sum_{i=a,b,c} [S_i]\{u_i\}. \tag{44}$$

The stresses are:

$$\begin{Bmatrix} \sigma_{xi} \\ \sigma_{yi} \\ \sigma_{xyi} \end{Bmatrix} = -\frac{Z_i}{I} \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} \qquad i = 1,2 \tag{45}$$

The principal stresses and their orientation are calculated in the same manner as those for the TRMEM element, section 4.87.4.6.

4.87.5.5 Stiffness Matrix Calculations for the TRPLT Element (Subroutine KTRPLT of Module SMA1).

The NASTRAN bending triangle (triangular plate element, TRPLT) is fabricated from three basic bending triangles. The geometry and notation are shown in Figure 5. The general approach is to calculate the stiffness matrices for all three subtriangles or basic triangles and use the constraint equation of equal slope at the midpoints of the connected edges to calculate a reduced stiffness matrix. Since only the partitions of the stiffness matrix related to one point (the pivot grid point) are used for each calculation, the extra partitions are not used. In the NASTRAN system, the basic bending triangle calculations are in subroutine form, and the variables necessary to call it are: x_b, x_c, y_c, the property and material coefficients, and the transformations for orienting the anisotropic materials. The following steps are used to calculate the overall stiffness matrix for the composite triangle.

1. The element coordinate system is defined by the location of the three grid points in basic coordinates, {x(1)}, {x(2)} and {x(3)}:

$$\{V_2\} = \{x(2)\} - \{x(1)\} , \tag{46}$$

$$\{V_3\} = \{x(3)\} - \{x(1)\} , \tag{47}$$

$$X_2 = |\{V_2\}| , \tag{48}$$

$$\{i\} = \frac{\{V_2\}}{X_2} , \tag{49}$$

$$Y_3 = |\{i\} \times \{V_3\}| , \tag{50}$$

$$\{k\} = \frac{\{i\} \times \{V_3\}}{Y_3} , \tag{51}$$

$$\{j\} = \{k\} \times \{i\} , \tag{52}$$

$$[E] = \left[\begin{array}{c|c|c} \{k\} & 0 & 0 \\ \hline 0 & \{i\} & \{j\} \end{array}\right] \cdot \quad (6 \times 3) \tag{53}$$

The locations of the points in this coordinate system are:

$$\{R(1)\} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} , \tag{54}$$

$$\{R(2)\} = \begin{Bmatrix} X_2 \\ 0 \end{Bmatrix} , \tag{55}$$

$$X_3 = \{V_3\}^T \{i\}, \tag{56}$$

$$\{R(3)\} = \begin{Bmatrix} X_3 \\ Y_3 \end{Bmatrix} , \tag{57}$$

$$\{R(4)\} = \frac{1}{3} \begin{Bmatrix} X_2 + X_3 \\ Y_3 \end{Bmatrix} . \tag{58}$$

2. For use in transferring points to the subtriangles, the integer matrix [M] is formed:

$$[M] = \begin{array}{c} \begin{array}{ccc} \text{Point a} & \text{Point b} & \text{Point c} \end{array} \\ \left[\begin{array}{ccc} 1 & 2 & 4 \\ 2 & 3 & 4 \\ 3 & 1 & 4 \end{array}\right] \end{array} \quad \begin{array}{c} \text{Subtriangle } \beta \\ \text{I} \\ \text{II} \\ \text{III} \end{array} \tag{59}$$

The Roman numerals I, II and III indicate the subtriangle numbers. Points 1, 2 and 3 are the corners of the whole triangle whose centroid is denoted by 4. Points a, b, and c are the corners of the subtriangles. Point c in the subtriangles is always the center point, 4. (see Figure 5).

Note: Steps 3 through 7 are performed for each subtriangle.

3. The location of the three points for each subtriangle, β, is defined by the vectors $\{r_i(a)\}$, $\{r_i(b)\}$, $\{r_i(c)\}$. In terms of the original vectors these are:

$$\left.\begin{aligned} r_i(a) &= R_i(M(\beta,1)) \\ r_i(b) &= R_i(M(\beta,2)) \\ r_i(c) &= R_i(4) \end{aligned}\right\} \qquad i = 1, 2 , \tag{60, 61, 62}$$

where $M(\beta, i)$ is the (β, i) element of the [M] matrix.

4. The variables necessary to calculate a basic bending triangle are x_b, x_c and y_c since the local coordinate system for each subtriangle is chosen such that the "a" point lies on the origin and the "b" point lies on the x axis.

For each triangle the following are calculated:

$$\ell = \sqrt{[r_1(b) - r_1(a)]^2 + [r_2(b) - r_2(a)]^2} \quad \text{(length of base)}, \tag{63}$$

$$w_1 = \frac{1}{\ell}(r_1(b) - r_1(a)), \tag{64}$$

$$w_2 = \frac{1}{\ell}(r_2(b) - r_2(a)) . \tag{65}$$

5. The matrix [T] used for transforming the element coordinates to subtriangle coordinates is formed:

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & w_1 & w_2 \\ 0 & -w_2 & w_1 \end{bmatrix}. \tag{66}$$

The material orientation angle, θ_m, is calculated for each subtriangle. The equations are:

$$\sin(\theta_m) = w_1 \sin(\theta) - w_2 \cos(\theta), \tag{67}$$

$$\cos(\theta_m) = w_1 \cos(\theta) + w_2 \sin(\theta). \tag{68}$$

The displacements in the subtriangle system are equal to [T] times the displacements in the original system (note det [T] is unit length, $x_b = R_1(2)$, $x_c = R_1(4)$ and $y_c = R_2(4)$ for subtriangle I).

6. The parameters x_b, x_c and y_c are then computed:

$$x_b = w_1[r_1(b) - r_1(a)] + w_2[r_2(b) - r_2(a)] , \tag{69}$$

$$x_c = w_1[r_1(c) - r_1(a)] + w_2[r_2(c) - r_2(a)] , \tag{70}$$

$$y_c = -w_2[r_1(c) - r_1(a)] + w_1[r_2(c) - r_2(a)]. \tag{71}$$

7. The stiffness matrices are formed as in the basic bending triangle (Equations 30 through 35) and give:

$$[k_{ia}], [k_{ib}], [k_{ic}], [k_{ca}] [k_{cb}], [k_{cc}] \quad i = \text{pivot grid point.}$$

They relate forces and displacements in the subtriangle coordinate system and are transformed to the overall element coordinate system (i.e., the same system as subtriangle I).

Since the stiffness matrices for each pivot grid point are calculated separately, not all of these partitions are used. For each pivot grid point, i, the following partitions are used:

$$[K_{i1}], [K_{i2}], [K_{i3}] ,$$

and

$$[K_{14}], [K_{24}], [K_{34}], [K_{44}] .$$

The integer mapping matrix [M] is used to determine if and where to add the partitions. The steps used for pivot point i and triangle β are:

a) $[T]^T [k_{ac}] [T]$ is added to $[K_{m,4}]$, $m = M(\beta,1)$

$[T]^T [k_{bc}] [T]$ is added to $[K_{m,4}]$, $m = M(\beta,2)$

$[T]^T [k_{cc}] [T]$ is added to $[K_{4,4}]$

b) If $M(\beta,1) = i$:

$[T]^T [k_{aa}] [T]$ is added to $[K_{ii}]$

$[T]^T [k_{ab}] [T]$ is added to $[K_{i\ell}]$, $\ell = M(\beta,2)$

or if $M(\beta,2) = i$

$[T]^T [k_{bb}] [T]$ is added to $[K_{ii}]$

$[T]^T [k_{ab}]^T [T]$ is added to $[K_{i\ell}]$, $\ell = M(\beta,1)$

c) The above is repeated for each of the three subtriangles.

8. The number 4 point in the middle is a dummy point, and since the displacements at point 4 are functions of the other displacements, it will be removed from the problem by including the calculations for the point 4 displacements in the calculations for the corner displacements, as shown in steps 8a, 8b, 8c and 8d.

a. Calculate the following geometric constants:

$$\ell_1 = \left(x_c^2 + y_c^2 \right)^{1/2} , \tag{72}$$

$$\ell_2 = \left| (x_b - x_c)^2 + y_c^2 \right|^{1/2} , \tag{73}$$

$$s_1 = \frac{x_c}{\ell_1} , \tag{74}$$

$$s_2 = \frac{x_b - x_c}{\ell_2} , \tag{75}$$

$$c_1 = \frac{y_c}{\ell_1} , \tag{76}$$

$$c_2 = \frac{y_c}{\ell_2} \quad . \tag{77}$$

b. The formulas for the locations of the midpoints are:

$$x_1 = \frac{1}{2} x_c \quad , \tag{78}$$

$$y_1 = \frac{1}{2} y_c \quad , \tag{79}$$

$$x_2 = \frac{x_b + x_c}{2} \quad , \tag{80}$$

$$y_2 = \frac{1}{2} y_c \quad . \tag{81}$$

c. The $[H_\psi]$ matrix is calculated as follows:

$$[H_\psi] = \begin{bmatrix} -2x_1 c_1 & x_1 s_1 - y_1 c_1 & 2y_1 s_1 & -3x_1^2 c_1 & y_1(2x_1 s_1 - y_1 c_1) & 3y_1^2 s_1 \\ 2x_2 c_2 & x_2 s_2 + y_2 c_2 & 2y_2 s_2 & 3x_2^2 c_2 & y_2(2x_2 s_2 + y_2 c_2) & 3y_2^2 s_2 \end{bmatrix} \tag{82}$$

The slopes in terms of the deflections of points a, b and c are defined by the matrices:

$$[\bar{H}_{\psi b} \mid \bar{H}_{\psi c}] = [H_\psi][H]^{-1} \tag{83}$$

$$[\bar{H}_{\psi a}] = - [H_\psi][H]^{-1} [B] + \begin{bmatrix} 0 & s_1 & c_1 \\ 0 & s_2 & c_2 \end{bmatrix} \tag{84}$$

where $[H]^{-1}$ and $[B]$ are calculated while generating the stiffness matrices. $[H]^{-1}$ is the inverse of the 6x6 matrix in Equation 25, and $[B] = \left[\frac{B_b}{B_c}\right]$ is a 6x3 matrix where $[B_b]$ and $[B_c]$ are calculated in Equations 27 and 28 respectively.

These are transformed to element coordinates by:

$$[H_{\psi\alpha}] = [\bar{H}_{\psi\alpha}] [T] \quad . \tag{85}$$

d. Each row of the matrix corresponds to the slope angle of the mid-point of the sides.

The first row defines the slope of the normal to the line connecting point "a" to the center point. The second row defines the slope of the normal to the line connecting point "b" to the center point. Using the $[H_{\psi\alpha}]$ matrices, four 3 by 3 matrices, $[G_M]$, are formed as follows:

The [M] matrix is now used:

$$[M] = \overbrace{\begin{matrix} a & b & c \end{matrix}}^{\alpha} \begin{matrix} \\ \\ \\ \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 4 \\ 3 & 1 & 4 \end{bmatrix} \begin{matrix} \text{I} \\ \text{II} \\ \text{III} \end{matrix} = \beta \ , \qquad (86)$$

For the $[H_{\psi\alpha}]^{\beta}$ matrix, (α = a, b and c; β = I, II or III), the number $M(\beta,\alpha)$ which identifies the matrix $[G_M]$ is found. The three terms in the first row of $[H_{\psi\alpha}]^{\beta}$ are added to the $M(\beta,1)$ row of the $[G_M]$ matrix. The three numbers in the second row of $[H_{\psi\alpha}]^{\beta}$ are added to the $M(\beta,2)$ row of the $[G_M]$ matrix. This procedure is repeated for the three $[H_{\psi\alpha}]^{\beta}$ matrices for each triangle β.

The stiffness matrix partitions of the whole plate are computed from:

$$[K^{e}_{ij}] = [K_{ij}] - ([G_4]^{-1}[G_i])^{T}[K_{i4}]^{T} - [K_{i4}][G_4]^{-1}[G_i]$$

$$+ ([G_4]^{-1}[G_i])^{T}[K_{44}][G_4]^{-1}[G_j] \qquad \begin{matrix} \text{for } i = 1, 2, 3 \\ j = 1, 2, 3 \end{matrix} . \qquad (87)$$

where $[K_{ij}]$ are computed as in step 7.

9. Using the locations of the three grid points in basic coordinates, the 3x3 transformation matrices $[T_j]$, j = 1, 2, 3, are calculated. The 6x6 matrices $[C_j]$ are formed as:

$$[C_j] = \left[\begin{array}{c|c} [T_j] & 0 \\ \hline 0 & [T_j] \end{array}\right] \qquad j = 1, 2, 3 \ . \qquad (88)$$

10. The stiffness matrix partitions in global coordinates for pivot point i are computed from:

$$[K^g_{ij}] = [C_i]^T[E][K^e_{ij}][E]^T[C_j] \quad (6x6) , \tag{89}$$

where:

$[K^e_{ij}]$ was calculated in step 8

$[E]$ was calculated in step 1

$[C_i]$, $[C_j]$ were calculated in step 9

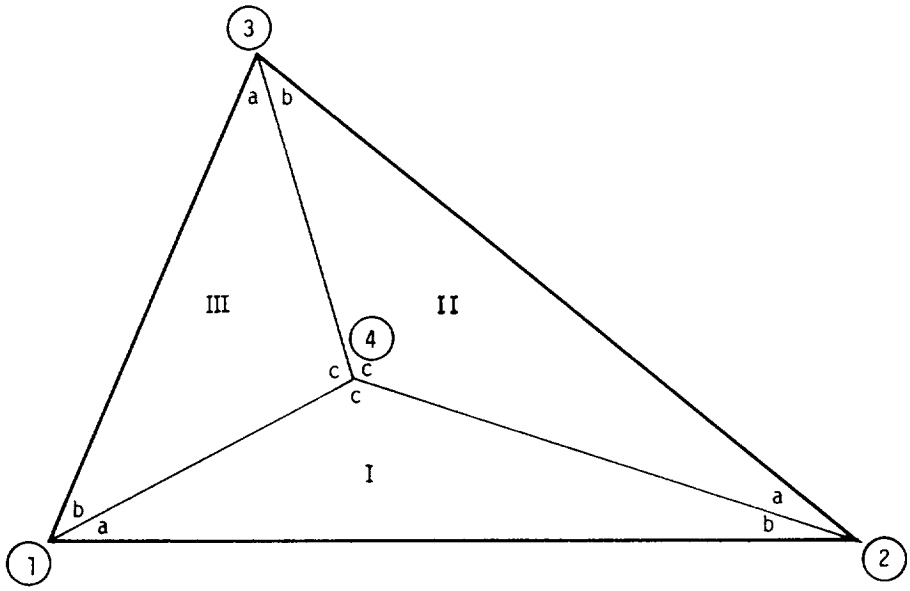


Figure 5. The triangular plate element, TRPLT.

1, 2, 3	=	GIVEN GRID POINTS
4	=	CENTROID OF TRIANGLE
I, II, III	=	BASIC SUBTRIANGLES
a, b, c	=	ORDERED VERTICES OF SUBTRIANGLES

4.87.5.6 Structural Damping Matrices for the TRPLT Element.

The structural damping matrices are:

$$[K^4_{ij}] = g_e[K^g_{ij}], \tag{90}$$

where g_e is the structural damping coefficient for the bending material referenced.

4.87.5.7 Stress and Element Force Calculations for the TRPLT Element (Subroutines STRPL1 and SBSPL2 of Module SDR2).

For stress recovery, the relationship between the center point and the corner points is used to describe the stress functions for each subtriangle. The stresses in each subtriangle at the center point are averaged to provide the final element stress and forces.

1. STRPL1 is used to calculate the phase 1 stress-displacement relations.

The following data are calculated using the same equations as those for the stiffness matrix generation routines.

$[C_i]$ - $i = 1, 2, 3$ - Global-to-basic transformations

$[E]$ - element to basic coordinate transformation

Values computed for each subtriangle:

- x_b, x_c, y_c - subtriangle point locations
- $\sin\theta$, $\cos\theta$ - material orientation
- w_1, w_2 - element-to-subtriangle coordinate transformation
- $[\bar{H}_{\psi a}]$, $[\bar{H}_{\psi b}]$, $[\bar{H}_{\psi c}]$ - normal slope angle relationship matrices

For each subtriangle, β = I, II, III, the following matrices are formed:

$$[T^{\beta}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & w_1 & w_2 \\ 0 & -w_2 & w_1 \end{bmatrix}, \tag{91}$$

$$[V^{\beta}] = \begin{bmatrix} w_1^2 & w_2^2 & -2w_1w_2 & 0 & 0 \\ w_2^2 & w_1^2 & 2w_1w_2 & 0 & 0 \\ w_1w_2 & -w_1w_2 & w_1^2-w_2^3 & 0 & 0 \\ 0 & 0 & 0 & w_1 & -w_2 \\ 0 & 0 & 0 & w_2 & w_1 \end{bmatrix} \tag{92}$$

For each subtriangle β, three 5x3 transformations are calculated, $[S_{\alpha}^{\beta}]$, a = a, b, c. These are transformed and added to the corresponding matrices for each point with the equation:

$$[S_M^*] = \frac{1}{3} \sum_{\beta} [V^{\beta}][S_{\alpha}^{\beta}][T^{\beta}] \ , \tag{93}$$

where the α, which denotes points on the subtriangle, corresponds to the grid point M on the overall triangle.

$$[H_{\psi\alpha}] = [\bar{H}_{\psi\alpha}] [T] \qquad \alpha = a, b, c \ . \tag{94}$$

The $[H_{\psi\alpha}]$ matrices are added to the corresponding $[G_M]$ matrix with the appropriate row interchanges. When the data for all three subtriangles have been generated, the following operations are performed:

$$[S_M^e] = [S_M^*] - [S_4^*][G_4]^{-1}[G_i] \tag{95}$$

$$\text{for } M = 1, 2, 3;$$

$$[S_M] = [S_M^e][E]^T[C_i] \tag{96}$$

$$\text{for } M = 1, 2, 3 \ .$$

2. Phase 2

a) The vector of forces is computed first.

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \\ V_x \\ V_z \end{Bmatrix} = \sum_{i=1}^{3} [S_i] \{u_i\}. \tag{97}$$

b) The stresses are then calculated from the equations

$$\left.\begin{aligned} \sigma_{xi} &= \frac{M_x Z_i}{I} \\ \sigma_{yi} &= \frac{M_y Z_i}{I} \\ \sigma_{xyi} &= \frac{M_{xy} Z_i}{I} \end{aligned}\right\} i = 1, 2. \tag{98, 99, 100}$$

The principal stresses and angles are calculated using the same formula as those for the membrane element (see section 4.87.4.6, Equations 28, 29 and 30).

4.87.5.8 Stiffness Matrix Calculations for the QDPLT Element (Subroutine KQDPLT of Module SMA1)

The quadrilateral plate element uses two sets of overlapping triangles as shown in Figure 6. The logic is the same as that for the quadrilateral membrane except that the order of the points of the triangles is chosen to place the triangle coordinate systems along the diagonals.

1. The following equations are used to calculate the three unit vectors, {i}, {j} and {k}, which define the element coordinate system.

$$\{V_i\} = \begin{Bmatrix} X_i \\ Y_i \\ Z_i \end{Bmatrix} , \quad i = 1, 2, 3, 4. \tag{101}$$

The diagonals are:

$$\{d_1\} = \{V_3\} - \{V_1\} , \tag{102}$$

$$\{d_2\} = \{V_4\} - \{V_2\} . \tag{103}$$

The area is calculated from:

$$A = \frac{1}{2} | (\{d_1\} \times \{d_2\}) | . \tag{104}$$

The normal to the plane is calculated from:

$$\{k\} = \frac{\{d_1\} \times \{d_2\}}{|\{d_1\} \times \{d_2\}|} , \tag{105}$$

$$\{a_1\} = \{V_2\} - \{V_1\} , \tag{106}$$

$$h = \{a_1\}^T \{k\} . \tag{107}$$

The vectors lying in the new plane are computed from:

$$\{i\} = \frac{\{a_1\} - h\{k\}}{|\{a_1\} - h\{k\}|} , \tag{108}$$

$$\{j\} = \{k\} \times \{i\} . \tag{109}$$

The nonzero positions of the points in the plane are computed from:

$$X_2 = \{a_i\}^T \{i\} \quad , \tag{110}$$

$$X_3 = \{d_i\}^T \{i\} \quad , \tag{111}$$

$$Y_3 = \{d_i\}^T \{j\} \quad , \tag{112}$$

$$X_4 = X_2 + (\{d_2\}^T \{i\}) \quad , \tag{113}$$

$$Y_4 = \{d_2\}^T \{j\} \quad , \tag{114}$$

$$\{R(i)\} = \begin{Bmatrix} X_i \\ Y_i \end{Bmatrix} \quad . \tag{115}$$

2. Element interior angle tests.

The interior angles of the quadrilateral must be less than 180°. The following checks accomplish this task.

	Test	Point with angle greater than 180°
If	$Y_4 < 0$	1
If	$Y_3 < 0$	2
If	$X_4 > X_2 - (X_2 - X_3) \frac{Y_4}{Y_3}$	3
If	$X_3 < \frac{Y_3}{Y_4} X_4$	4

3. The relative data for each subtriangle must be calculated and passed to the matrix calculation subroutine. The integer mapping matrix [M] denotes which points, 1, 2, 3, and 4 of the quadrilateral, are used in the subtriangle. The row position indicates the subtriangle to which the point belongs, and the column position indicates the corresponding point in that subtriangle.

$$[M] = \begin{array}{c} \begin{array}{ccc} \underline{\text{Point a}} & \underline{\text{Point b}} & \underline{\text{Point c}} \end{array} \\ \begin{bmatrix} 2 & 4 & 1 \\ 3 & 1 & 2 \\ 4 & 2 & 3 \\ 1 & 3 & 4 \end{bmatrix} \end{array} \quad \begin{array}{c} \underline{\text{Triangle No.}} \\ \text{I} \\ \text{II} \\ \text{III} \\ \text{IV} \end{array}$$

4. For each triangle the stiffness matrix is calculated in its own coordinate system. This system has its origin at point a, point b lies on the x axis, and point c has a positive y component. $\{R(i)\}$ values are transferred to $\{r(\alpha)\}$ values (α = a, b, c), and the following calculations are performed:

$$\{v(b)\} = \{r(b)\} - \{r(a)\}, \tag{116}$$

$$\{v(c)\} = \{r(c)\} - \{r(a)\}, \tag{117}$$

$$\begin{Bmatrix} w_1 \\ w_2 \end{Bmatrix} = \frac{\{v(b)\}}{|\{v(b)\}|}, \tag{118}$$

$$x_b = |\{v(b)\}|, \tag{119}$$

$$\begin{Bmatrix} x_c \\ y_c \end{Bmatrix} = \begin{bmatrix} w_1 & w_2 \\ -w_2 & w_1 \end{bmatrix} \{v(c)\}. \tag{120}$$

x_b, x_c and y_c are the point locations needed by the subroutine.

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & w_1 & w_2 \\ 0 & -w_2 & w_1 \end{bmatrix}. \tag{121}$$

w_1 is the x component of the new x axis and w_2 is its y component, [T] transforms the z displacement and the two angles from the quadrilateral system to the triangle system.

In order to calculate the material matrices in the basic triangle routine, the material orientation angle, θ_m, is :

$$\sin\theta_m = w_1 \sin\theta - w_2 \cos\theta \ , \tag{122}$$

$$\cos\theta_m = w_1 \cos\theta + w_2 \sin\theta \ . \tag{123}$$

w_1 and w_2 are the cosine and sine of the angle made between the base of the triangle and the material orientation axis.

5. The output of the basic bending triangle routine are the 3x3 matrices:

$$[k_{aa}],\ [k_{ab}],\ [k_{bb}],\ [k_{ac}],\ [k_{bc}],\ [k_{cc}] \ .$$

To transform these to the quadrilateral system the following equation is used:

$$[K^e_{ij}] = \frac{1}{2}[T]^T[k_{ij}][T] \ . \tag{124}$$

These matrices are added to the current positions in the quadrilateral matrix partitions using the [M] matrix in step 3.

6. For each pivot point i the following 3x3 partitions are formed:

$$[k^e_{ij}],\ \text{for } j = 1, 2, 3, 4 \ .$$

7. Using the geometry data, the 3x3 global-to-basic transformations $[T_j]$ are formed for j = 1, 2, 3, 4. These are expanded to the 6x6 matrices $[C_j]$:

$$[C_j] = \left[\begin{array}{c|c} T_j & 0 \\ \hline 0 & T_j \end{array}\right] . \qquad (125)$$

8. The stiffness matrix partitions in global coordinates are found from:

$$[K^g_{ij}] = [C_i]^T[E][k^e_{ij}][E]^T[C_j] , \qquad (126)$$

where [E] is defined in Equation 53.

4.87.5.9 Stress and Element Force Calculations for QDPLT Element (Subroutines SQDPL1 and SBSPL2 of Module SDR2)

1. SQDPL1 calculates the phase 1 stress-displacement relations. A coordinate system matrix [E], the subtriangle base vectors $\{w_1, w_2\}^T$ and the global-to-basic transformation matrices are calculated with the same equations as those used in the plate element stiffness matrix calculations. For each subtriangle, β, the following matrices are formed:

$$[T^\beta] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & w_1 & w_2 \\ 0 & -w_2 & w_1 \end{bmatrix} , \qquad (127)$$

$$[V^\beta] = \begin{bmatrix} w_1^2 & w_2^2 & -2w_1w_2 & 0 & 0 \\ w_2^2 & w_1^2 & 2w_1w_2 & 0 & 0 \\ w_1w_2 & -w_1w_2 & (w_1^2-w_2^2) & 0 & 0 \\ 0 & 0 & 0 & w_1 & -w_2 \\ 0 & 0 & 0 & w_2 & w_1 \end{bmatrix} . \qquad (128)$$

For each subtriangle the stresses must be calculated at the intersection point of the diagonals. In the quadrilateral system:

$$x_5 = \frac{x_2 x_3 y_4}{x_3 y_4 + (x_2 - x_4) y_3} , \tag{129}$$

$$y_5 = \frac{y_3}{x_3} x_5 . \tag{130}$$

For each subtriangle:

$$\{V_5\} = \{r(a)\} - \begin{Bmatrix} x_5 \\ y_5 \end{Bmatrix} ; \quad x_q = |\{V_5\}|; \quad y_q = 0; \tag{131}$$

where x_q and y_q denote the location of the intersection of the diagonals of the quadrilateral in the subtriangle coordinate system. The stresses are calculated at this point.

$[T^\beta]$ transforms the translation and two rotations in the element system to the subtriangle system. $[V^\beta]$ transforms the three moments and two shears of the subtriangle to the element system. For each subtriangle β, three 5x3 transformations are calculated, $[S_\alpha^\beta]$, α = a, b, c. These are transformed and added to the corresponding matrices for each point with the equation:

$$[S_M^e], = \frac{1}{4} \sum_\beta [V^\beta][S_\alpha^\beta][T^\beta] , \tag{132}$$

where the α, which denotes points on the subtriangle, corresponds to the grid point M on the quadrilateral.

2. Using the basic-to-element and global-to-basic transformations, the stress matrices $[S_M]$ in global coordinates are formed from:

$$[S_M] = [S_M^e][E]^T \left[\begin{array}{c|c} [T_i] & 0 \\ \hline 0 & [T_i] \end{array}\right] . \tag{133}$$

3. The additional data for phase 2 calculations are

I, Z_1, Z_2 from the ECPT data.

4. Phase 2 (Subroutine SBSPL2)

   The 5 by 6 $[S_M]$ matrices are used in the same manner as the TRPLT elements (see Equations 97 through 100).

4.87.5.10 Lumped Mass Matrix Generation for the TRBSC, TRPLT, and QDPLT Elements (Subroutine MASSTQ of Module SMA2).

The lumped mass matrices are calculated in the same manner as the triangular or quadrilateral membrane except that the material density is not used (see Equations 86 through 96 in section 4.87.4.10). The only contribution to the mass matrix from these elements is the nonstructural mass, μ.

4.87.5.11 Coupled Mass Matrix Calculations for the TRBSC Element (Subroutine MTRBSC of Module SMA2).

The mass properties of the two-dimensional elements are defined by their geometry, the mass density given by the material, the thickness of the element and the nonstructural mass. The normal execution of NASTRAN will result in the calculation of the total mass of the element and distribute it as lumped masses at the connected grid points (subroutine MASSTQ of module SMA2). If the parameter CØUPBAR is set by the user, the elements with bending properties will have their mass distributed according to their elastic properties. This results in element mass matrices with directional properties and coupling terms between points in an element. Since the thickness of the TRBSC element is zero, a coupled mass matrix for this element does not exist. The MTRBSC subroutine is used exclusively by subroutines MTRPLT and MQDPLT.

1. The mass matrix $[\tilde{M}]$ in generalized coordinates is calculated in the following steps.

   a. Integral values I_{ij} used in the mass matrices are calculated from the formulas:

$$A_{0j} = \left[\frac{x_b}{j+1} - \frac{x_b - x_c}{j+2} \right] y_c^{j+1} \quad , \qquad j = 0,1,\ldots 6 \tag{134}$$

$$A_{ij} = \frac{(x_c)^{i+1}\,(y_c)^{j+1}}{(i+1)\,(i+j+2)} + \frac{i x_b}{(i+j+2)} A_{i-1,j} \quad , \qquad \begin{matrix} i = 1,2,\ldots 6 \\ j = 0,1,\ldots 6 \end{matrix} \tag{135}$$

$$B_{ij} = -\frac{(x_c)^{i+1}}{(y_c)^{i+1}\,(i+1)(i+j+2)} (y_c)^{i+j+2} \quad , \qquad \begin{matrix} i = 0,1,\ldots 6 \\ j = 0,1,\ldots 6 \end{matrix} \tag{136}$$

$$I_{ij} = m[A_{ij} + B_{ij}] \quad , \qquad \begin{matrix} i = 0,1,\ldots 6 \\ j = 0,1,\ldots 6 \end{matrix} \tag{137}$$

where m is the nonstructural mass, and where x_b, x_c and y_c are computed in Equations 7, 8, and 9 respectively.

b. Partition $[\tilde{M}]$ into submatrices.

$$[\tilde{M}] \Longrightarrow \left[\begin{array}{c|c} \overline{M}_{aa} & M_{ar} \\ \hline M_{ar}^T & M_{rr} \end{array}\right] , \tag{138}$$

where $[\overline{M}_{aa}]$ is a 3 by 3 matrix, $[M_{ar}]$ is a 3 by 6 matrix and $[M_{rr}]$ is a 6 by 6 matrix.

c. The mass matrix $[\overline{M}_{aa}]$ is given by:

$$[\overline{M}_{aa}] = \begin{bmatrix} I_{00} & I_{01} & -I_{10} \\ I_{01} & I_{02} & -I_{11} \\ -I_{10} & -I_{11} & I_{20} \end{bmatrix} . \tag{139}$$

d. The other matrices, $[M_{ar}]$ and $[M_{rr}]$, are less simple. The algebraic expressions for the elements of these matrices are given in Tables 1a and 1b below.

Table 1a. Elements of the 3 by 6 Matrix $[M_{ar}]$.

Notes: $H_{\gamma q_{ij}}$ is contracted to H_{ij}, where $[H_{\gamma q}]$ is computed in Equation 22.

$M_{ar_{ij}}$ is contracted to M_{ik}, where M_{ik} represents an element of $[\tilde{M}]$ given in Equation 138.

$M_{14} = I_{20}$

$M_{15} = I_{11}$

$M_{16} = I_{02}$

$M_{17} = I_{30} + H_{14}I_{10} + H_{24}I_{01}$

Table 1a (con'd). Elements of the 3 by 6 Matrix $[M_{ar}]$.

$$M_{18} = I_{12} + H_{15}I_{10} + H_{25}I_{01}$$

$$M_{19} = I_{03} + H_{16}I_{10} + H_{26}I_{01}$$

$$M_{24} = I_{21}$$

$$M_{25} = I_{12}$$

$$M_{26} = I_{03}$$

$$M_{27} = I_{31} + H_{14}I_{11} + H_{24}I_{02}$$

$$M_{28} = I_{13} + H_{15}I_{11} + H_{24}I_{02}$$

$$M_{29} = I_{04} + H_{16}I_{11} + H_{26}I_{02}$$

$$M_{34} = -I_{30}$$

$$M_{35} = -I_{21}$$

$$M_{36} = -I_{12}$$

$$M_{37} = -I_{40} - H_{14}I_{20} - H_{24}I_{11}$$

$$M_{38} = -I_{22} - H_{15}I_{20} - H_{25}I_{11}$$

$$M_{39} = -I_{13} - H_{16}I_{20} - H_{26}I_{11}$$

Table 1b. Elements of the 6 by 6 Matrix $[M_{rr}]$.

Notes: $H_{YQ_{ij}}$ is contracted to H_{ij}; $M_{rr_{ij}}$ is contracted to M_{ik}, where M_{ik} represents an element of $[\tilde{M}]$ given in Equation 138; $M_{ij} = M_{ji}$.

$$M_{44} = I_{40}$$

$$M_{45} = I_{31}$$

$$M_{46} = I_{22}$$

$$M_{47} = I_{50} + H_{14}I_{30} + H_{24}I_{21}$$

$$M_{48} = I_{32} + H_{15}I_{30} + H_{25}I_{21}$$

$$M_{49} = I_{23} + H_{16}I_{30} + H_{26}I_{21}$$

$$M_{55} = I_{22}$$

$$M_{56} = I_{13}$$

$$M_{57} = I_{41} + H_{14}I_{21} + H_{24}I_{12}$$

$$M_{58} = I_{23} + H_{15}I_{21} + H_{25}I_{12}$$

$$M_{59} = I_{14} + H_{16}I_{21} + H_{26}I_{12}$$

$$M_{66} = I_{04}$$

$$M_{67} = I_{32} + H_{14}I_{12} + H_{24}I_{03}$$

$$M_{68} = I_{14} + H_{15}I_{12} + H_{25}I_{03}$$

$$M_{69} = I_{05} + H_{16}I_{12} + H_{26}I_{03}$$

$$M_{77} = I_{60} + 2H_{14}I_{40} + 2H_{24}I_{31} + (H_{14})^2I_{20} + 2H_{14}H_{24}I_{11} + (H_{24})^2I_{02}$$

Table 1b (con'd). Elements of the 6 by 6 Matrix $[M_{rr}]$.

$$M_{78} = I_{42} + H_{15}I_{40} + H_{25}I_{31} + H_{14}I_{22} + H_{14}H_{15}I_{20} + H_{14}H_{25}I_{11} + H_{24}I_{13} + H_{24}H_{15}I_{11} + H_{24}H_{25}I_{02}$$

$$M_{79} = I_{33} + H_{16}I_{40} + H_{26}I_{31} + H_{14}I_{13} + H_{14}H_{16}I_{20} + H_{14}H_{26}I_{11} + H_{24}I_{04} + H_{24}H_{16}I_{11} + H_{24}H_{26}I_{02}$$

$$M_{88} = I_{24} + 2H_{15}I_{22} + 2H_{25}I_{13} + (H_{15})^2I_{20} + 2H_{15}H_{25}I_{11} + (H_{25})^2I_{02}$$

$$M_{89} = I_{15} + H_{16}I_{22} + H_{26}I_{13} + H_{15}I_{13} + H_{15}H_{16}I_{20} + H_{15}H_{26}I_{11} + H_{25}I_{04} + H_{25}H_{16}I_{11} + H_{25}H_{26}I_{02}$$

$$M_{99} = I_{06} + 2H_{16}I_{13} + 2H_{26}I_{04} + (H_{16})^2I_{20} + 2H_{16}H_{26}I_{11} + (H_{26})^2I_{02}$$

2. The mass matrix $[M_{mass}]$ in element coordinates is calculated from the following equation:

$$[M_{mass}] = \left[\begin{array}{c|c} I & 0 \\ \hline -H^{-1}B & H^{-1} \end{array}\right]^T \left[\begin{array}{c|c} \bar{M}_{aa} & M_{ar} \\ \hline M_{ar}^T & M_{rr} \end{array}\right] \left[\begin{array}{c|c} I & 0 \\ \hline -H^{-1}B & H^{-1} \end{array}\right] , \quad (140)$$

where $[I]$ is a 3 x 3 identity matrix, $[H]^{-1}$ is calculated as in Equation 25, $[B] = \left[\begin{array}{c} B_b \\ \hline B_c \end{array}\right]$ is calculated as in Equations 27 and 28.

The calculations are broken down into the following steps where:

$$[M_{mass}] = \left[\begin{array}{c|c|c} M_{aa} & M_{ab} & M_{ac} \\ \hline M_{ba} & M_{bb} & M_{bc} \\ \hline M_{ca} & M_{cb} & M_{cc} \end{array}\right] , \quad (141)$$

and $[M_{ij}]$, i = a, b, c and j = a, b, c are 3 by 3 matrices.

a) Compute:

$$[M] = [H^{-1}]^T [M_{rr}] [H^{-1}] . \quad (142)$$

b) Partition:

$$[M] \Longrightarrow \left[\begin{array}{c|c} M_{bb} & M_{bc} \\ \hline M_{cb} & M_{cc} \end{array}\right] . \quad (143)$$

c) Compute:

$$[M_{ai}] = [M_{ar}] [H^{-1}] . \quad (144)$$

d) Partition:

$$[M_{ai}] \Longrightarrow [\bar{M}_{ab} \mid \bar{M}_{ac}] . \quad (145)$$

e) Calculate:

$$[M_{ab}] = [\bar{M}_{ab}] - [B_b]^T[M_{bb}] - [B_c]^T[M_{cb}] \quad , \tag{146}$$

$$[M_{ac}] = [\bar{M}_{ac}] - [B_b]^T[M_{bc}] - [B_c]^T[M_{cc}] \quad , \tag{147}$$

$$[M_{aa}] = [\bar{M}_{aa}] - [B_b]^T[M_{ab}]^T - [B_c]^T[M_{ac}]^T - [\bar{M}_{ab}][B_b] - [\bar{M}_{ac}][B_c] \quad , \tag{148}$$

$$[M_{ba}] = [M_{ab}]^T \quad , \tag{149}$$

$$[M_{ca}] = [M_{ac}]^T \quad . \tag{150}$$

4.87.5.12 Mass Matrix Calculations for the TRPLT Element (Subroutine MTRPLT of Module SMA2)

1. The general calculations for the mass matrix of the triangular plate element, TRPLT, are the same as those for the stiffness matrix calculations (See Equations 46 through 71).

2. For each subtriangle the output from the basic bending triangle subroutine are the nine 3 x 3 matrices given in Equation 141:

$$\begin{bmatrix} M_{aa} & M_{ab} & M_{ac} \\ M_{ba} & M_{bb} & M_{bc} \\ M_{ca} & M_{cb} & M_{cc} \end{bmatrix} .$$

They relate forces and accelerations in the subtriangle coordinate system and must be transformed to the overall element coordinate system (i.e., the same system as subtriangle I).

The matrix partitions in the subtriangles are added to the correct matrix partition for the whole triangle. For example, for subtriangle number II, $[M_{aa}]$ is transformed and added to $[\bar{M}_{22}]$, $[M_{ab}]$ is transformed and added to $[\bar{M}_{23}]$, $[M_{ac}]$ is transformed and added to $[\bar{M}_{24}]$, $[M_{ba}]$ is transformed and added to $[\bar{M}_{32}]$, etc.

Since the mass matrices for each pivot grid point are calculated separately, not all of these partitions are used. For each pivot grid point, i, the matrices which are used will be:

$$[\bar{M}_{i1}],\ [\bar{M}_{i2}],\ [\bar{M}_{i3}] \quad ,\ i = \text{point 1, 2 or 3 of composite triangle}$$

and

$$[\bar{M}_{14}],\ [\bar{M}_{24}],\ [\bar{M}_{34}],\ [\bar{M}_{44}] \quad .$$

3. The number 4 point in the middle is a dummy point and is removed from the problem in the same manner as in the computation of the stiffness matrix (see Equations 72 through 86).

The mass matrix partitions of the whole plate are:

$$[M^e_{ij}] = [\bar{M}_{ij}] - ([G_4]^{-1}[G_i])^T[\bar{M}_{j4}]^T - [\bar{M}_{i4}][G_4]^{-1}[G_j] \qquad (151)$$

$$+ ([G_4]^{-1}[G_i])^T\,[\bar{M}_{44}]\,[G_4]^{-1}\,[G_j] \qquad \begin{matrix} i = 1, 2, 3 \\ j = 1, 2, 3 \end{matrix} \quad .$$

Notice that if i and j were interchanged the matrix would be transposed; this indicates that the whole mass matrix is symmetric.

4. Using the locations of the three grid points in basic coordinates, calculate the 3 x 3 transformation matrices $[T_j]$, j = 1, 2, 3. Form the 6 x 6 matrices:

$$[C_j] = \left[\begin{array}{c|c} T_j & 0 \\ \hline 0 & T_j \end{array}\right] \quad . \qquad (152)$$

5. The 3 x 3 mass matrix partitions are expanded to 6x6 size and transformed to global coordinates using the logic:

$$M_3 = \frac{m}{6} X_2 Y_3 \quad , \qquad (153)$$

where m is the nonstructural mass, and X_2, Y_3 are the x-coordinate and y-coordinate of points 2 and 3 respectively.

$$[\bar{M}^e_{ij}] = \begin{bmatrix} M_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & M_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & M_{11} & M_{12} & M_{13} & 0 \\ 0 & 0 & M_{21} & M_{22} & M_{23} & 0 \\ 0 & 0 & M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} , \quad (154)$$

for i = j = 1, 2 or 3; and

$$[\bar{M}^e_{ij}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & M_{11} & M_{12} & M_{13} & 0 \\ 0 & 0 & M_{21} & M_{22} & M_{23} & 0 \\ 0 & 0 & M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} , \quad (155)$$

for $i \neq j$.

The mass matrix partitions in global coordinates are:

$$[M^g_{ij}] = [C_i]^T[E][\bar{M}^e_{ij}][E]^T[C_j] , \quad (156)$$

where

$$[C_i] = \left[\begin{array}{c|c} T_i & 0 \\ \hline 0 & T_i \end{array}\right] \quad (157)$$

is the global-to-basic transformation matrix,

and

$$[E] = \left[\begin{array}{c|c|c|c|c|c} \{i\} & \{j\} & \{k\} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \{i\} & \{j\} & \{k\} \end{array}\right] \qquad (158)$$

is the 6 x 6 element-to-basic transformation matrix.

4.87.5.13 Mass Matrix Calculations for the QDPLT Element (Subroutine MQDPLT of Module SMA2).

1. The general calculations for the mass matrix of the quadrilateral plate element, QDPLT, are the same as those for the stiffness matrix calculations (see Equations 101 through 123).

2. For each subtriangle, the output from the basic bending triangle subroutine are the nine 3 x 3 matrices:

$$\left[\begin{array}{c|c|c} M_{aa} & M_{ab} & M_{ac} \\ \hline M_{ba} & M_{bb} & M_{bc} \\ \hline M_{ca} & M_{cb} & M_{cc} \end{array}\right] .$$

These are transformed to the quadrilateral system by the following equation:

$$[\bar{M}_{ij}] = \frac{1}{2} [T]^T [M_{ij}] [T] \quad , \qquad (159)$$

where [T] is given in Equation 121.

These matrices are added to their corresponding positions in the quadrilateral matrix partitions $[M^e_{ij}]$, in the same manner as that for the stiffness matrix partitions of the quadrilateral (See step 3 of Section 4.87.5.8).

3. For each pivot point i, the following 3 x 3 partitions are formed:

$$[M^e_{ij}], \text{ for } j = 1, 2, 3, 4 \quad .$$

4. The mass at each point in the plane of the element is due to the mass of the attached triangles. The masses of the triangles are:

$$m^{\beta} = \frac{m}{4} x_b^{\beta} y_c^{\beta} \quad , \qquad \beta = \text{I, II, III and IV} \quad , \tag{160}$$

where m is the nonstructural mass, and x_b^{β} and y_c^{β} are the x-coordinate and y-coordinate of points b and c respectively of subtriangle β.

The masses at the points are:

$$m_1 = \frac{1}{3} [m^{I} + m^{II} + m^{IV}] \quad , \tag{161}$$

$$m_2 = \frac{1}{3} [m^{I} + m^{II} + m^{III}] \quad , \tag{162}$$

$$m_3 = \frac{1}{3} [m^{II} + m^{III} + m^{IV}] \quad , \tag{163}$$

$$m_4 = \frac{1}{3} [m^{I} + m^{III} + m^{IV}] \quad . \tag{164}$$

5. The 3 x 3 mass matrix partitions are expanded to 6 x 6 matrices and the in-plane mass effects are added using the logic:

$$m_z^i = \frac{m_i h}{2} \quad , \tag{165}$$

$$I_z^i = \frac{m_i h^2}{4} \quad , \tag{166}$$

$$[\bar{M}^e_{ij}] = \begin{bmatrix} m_i & 0 & 0 & 0 & -m^i_z & 0 \\ 0 & m_i & 0 & m^i_z & 0 & 0 \\ 0 & 0 & M_{11} & M_{12} & M_{13} & 0 \\ 0 & m^i_z & M_{21} & M_{22} + I^i_z & M_{23} & 0 \\ -m^i_z & 0 & M_{31} & M_{32} & M_{33} + I^i_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (167)$$

for i = j = 1 or 3; and

$$[\bar{M}^e_{ij}] = \begin{bmatrix} m_i & 0 & 0 & 0 & m^i_z & 0 \\ 0 & m_i & 0 & -m^i_z & 0 & 0 \\ 0 & 0 & M_{11} & M_{12} & M_{13} & 0 \\ 0 & -m^i_z & M_{21} & M_{22} + I^i_z & M_{23} & 0 \\ m^i_z & 0 & M_{31} & M_{32} & M_{33} + I^i_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (168)$$

for i = j = 2 or 4; and

$$[\bar{M}^e_{ij}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & M_{11} & M_{12} & M_{13} & 0 \\ 0 & 0 & M_{21} & M_{22} & M_{23} & 0 \\ 0 & 0 & M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} , \quad (169)$$

for $i \neq j$.

6. Using the geometry data, the 3 x 3 global-to-basic transformations $[T_j]$ are formed for $j = 1, 2, 3, 4$. These are expanded to 6 x 6 matrices:

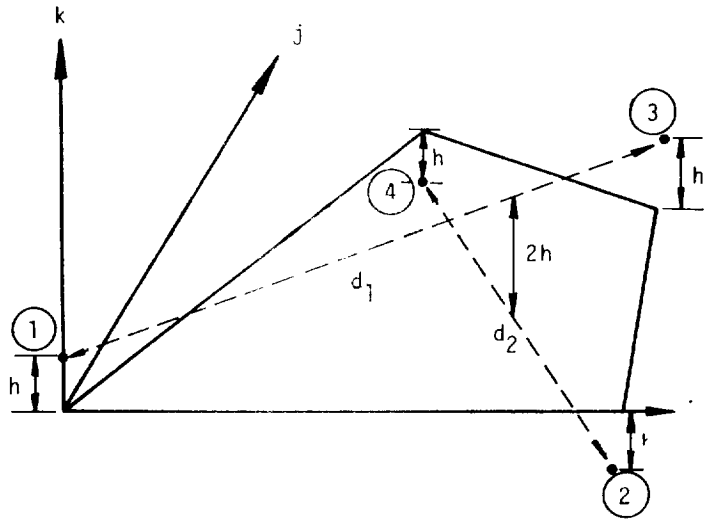
$$[C_j] = \left[\begin{array}{c|c} T_j & 0 \\ \hline 0 & T_j \end{array}\right] . \quad (170)$$

7. The 6 x 6 element-to-basic transformation matrix is:

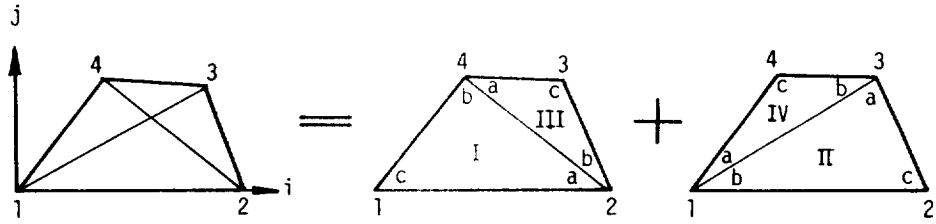
$$[E] = \left[\begin{array}{c|c|c|c|c|c} \{i\} & \{j\} & \{k\} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \{i\} & \{j\} & \{k\} \end{array}\right] . \quad (171)$$

8. The 6 x 6 matrices are transformed to global coordinates using the equation:

$$[M^g_{ij}] = [C_i]^T [E] [\bar{M}^e_{ij}] [E]^T [C_j] . \quad (172)$$



a) Definition of plane



b) Subtriangles

Figure 6. Geometry of the quadrilateral plate element, QDPLT.

4.87.6 The TRIA1, TRIA2, QUAD1 and QUAD2 Elements

These elements have the properties of both membranes and bending plates. The TRIA1 and QUAD1 elements are triangles and quadrilaterals which may use separate thicknesses and materials for membrane, bending and transverse shear action. The TRIA2 and QUAD2 elements are triangles and quadrilaterals which use one thickness and one material to simulate a homogeneous plate with consistent membrane, bending and transverse shear properties.

If these elements use anisotropic materials (defined on a MAT2 bulk data card), the material is oriented with respect to the element coordinate system. The definition of the coordinate system is as follows: the vector from the first point to the second point defines the base or x axis. The z axis is normal to the average plane of the elements, and the third and fourth points have positive y values.

Mass matrices, thermal loads, and differential stiffness matrices for these elements use only the membrane properties.

4.87.6.1 Input Data for the TRIA1, TRIA2, QUAD1 and QUAD2 Elements

1. The ECPT/EST entries for the TRIA1 or QUAD1 elements are:

Symbol	Description
SIL_i, $i = 1,2,3$ or $i = 1,2,3,4$	Scalar indices of the connected grid points.
N_i, X_i, Y_i, Z_i ; $i = 1,2,3$ or $i = 1,2,3,4$	Referenced local coordinate system and location in basic coordinates of connected grid points.
θ	Material orientation angle.
MAT ID_m	Material identification number for membrane properties.
t_m	Membrane thickness.
MAT ID_b	Material identification number for bending properties.

Symbol	Description
I	Bending inertia
MAT ID_s	Material identification number for transverse shear properties.
t_s	Transverse shear thickness
μ	Nonstructural mass per area
Z_1, Z_2	Outer fiber distances for stress calculations.
t_μ	Temperature for material properties

2. ECPT Entries for the TRIA2 or QUAD2 Elements.

Symbol		Description
SIL_i	$i = 1,2,3$ or $i = 1,2,3,4$	Scalar indices of connected grid points.
N_i, X_i, Y_i, Z_i	$i = 1,2,3$, or $i = 1,2,3,4$	Referenced local coordinate system and location vector in basic coordinates of connected grid points.
θ		Material orientation angle.
MAT ID		Material identification number
t_m		Element thickness
μ		Nonstructural mass per area.
t_μ		Temperature for material properties

4.87.6.2 Stiffness Matrix Calculations (Subroutine KTRIOD of Module SMA1).

The TRIA1 or QUAD1 element ECPT data are rearranged to correspond to the (TRMEM or QDMEM) membrane ECPT form, and the routine of the TRMEM or QDMEM element is used. The ECPT data are then rearranged to correspond to the ECPT data of a TRPLT or QDPLT element and the respective plate routine is used. Each routine is entirely independent.

The TRIA2 and QUAD2 elements are treated in the same manner except that the arrangement of the ECPT data is different. The type "2" element uses the single material for all three material properties of the type "1" element. The membrane and transverse shear thickness equal the single thickness of the type "2" element. The bending inertia, I_b, for the plate property is:

$$I_b = \frac{t^3}{12} \tag{1}$$

4.87.6.3 Lumped Mass Matrix Generation (Subroutine MASSTQ of Module SMA2)

The bending properties are disregarded for the lumped mass matrix calculations, and the element mass matrices are computed exactly as the ones for the TRMEM and QDMEM elements.

4.87.6.4 Thermal Load Calculations (Subroutine EDTL of Module SSG1)

The bending properties are disregarded for the thermal load calculations, and the thermal load vectors are computed exactly as the ones for the TRMEM and QDMEM elements.

4.87.6.5 Element Stress and Force Calculations (Subroutines STRQD1 and STRQD2 of Module SDR2)

As with the stiffness matrix calculations, the data are rearranged and the stresses for both the membrane and plate deformations are calculated. The element forces are calculated for the plate only.

1. For the TRIA2 and QUAD2 elements the outer fiber distances Z_1 and Z_2 are:

$$Z_1 = \frac{t}{2} \tag{2}$$

$$Z_2 = -\frac{t}{2} \tag{3}$$

The membrane and plate stresses are added together as follows for Z_1:

$$\begin{Bmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{Bmatrix}_{\text{composite}} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix}_{\text{membrane}} + \begin{Bmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{Bmatrix}_{\text{bending}}, \tag{4}$$

and for Z_2,

$$\begin{Bmatrix} \sigma_{x2} \\ \sigma_{y2} \\ \sigma_{xy2} \end{Bmatrix}_{composite} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix}_{membrane} + \begin{Bmatrix} \sigma_{x2} \\ \sigma_{y2} \\ \sigma_{xy2} \end{Bmatrix}_{bending} . \quad (5)$$

2. The principal stresses and their orientation are calculated from the above results, as in Equations 28, 29 and 30 of section 4.87.4.6.

4.87.6.6 Differential Stiffness Matrix Calculations

The element differential stiffness matrices are formed using the membrane properties only. The element differential stiffness matrix routines, DTRMEM and DQDMEM, are called directly from subroutine DS1A of module DSMG1 (see sections 4.87.4.7 and 4.87.4.13).

4.87.6.7 Coupled Mass Matrix Calculations (Subroutine MTRIQD of Module SMA2).

In the lumped mass case these elements are processed using the membrane mass calculation routines (subroutine MASSTQ of module SMA2). When coupled mass is requested, the plate mass calculations will be used instead. The ECPT data are rearranged to the appropriate TRPLT or QDPLT format, and the respective plate routine is used. The mass per area is now calculated using the material mass density ρ and the thickness t_m for the membrane definition of the element and added to the nonstructural mass:

$$m = \mu + \rho\, t_m \quad . \quad (6)$$

m is now used instead of μ for the plate calculations.

4.87.6.8 Piecewise Linear Analysis Calculations (Subroutines PSTRI1, PSTRI2, PSQAD1, and PSQAD2 of Module PLA3 and PKTRI1, PKTRI2, PKQAD1 and PKQAD2 of Module PLA4)

The additional ECPTNL and ESTNL entries are:

ε_o^* - The previously computed strain value once removed.

ε^* - The previously computed strain value.

E^* - The previously computed modulus of elasticity.

σ_x^*, σ_y^*, σ_{xy}^* } The previously computed membrane stresses.

M_x^*, M_y^*, M_{xy}^*, V_x^*, V_y^* } The previcusly computed element forces (ESTNL only).

All of the above values are initially zero with the exception of E^*, which is initially the original modulus of elasticity present on a MAT1 card.

In module PLA3, the incremental element stress matrix is calculated by first rearranging the ESTNL data to correspond to the ESTNL data for a TRMEM or QDMEM, and then the membrane stresses are calculated in the same manner as Equations 103 through 105 of section 4.87.4.14. Then the ESTNL data are rearranged to correspond to the EST data for a TRPLT (or QDPLT) and the incremental bending forces for the TRPLT (or QDPLT) element are calculated in the same manner as in Equation 97 of section 4.87.5.7. However, if the bending material properties are the same as the membrane material properties, then the 3 by 3 bending material properties matrix ($[G_b]$ in Equation 11 of section 4.87.5.2) is replaced by the matrix given in Equation 97 of section 4.87.4.14. In addition the displacement vector $\{u_1\}$ in Equation 44 of section

4.87.5.4 (or Equation 97 of section 4.87.5.7) are replaced by an incremental displacement vector $\{\Delta u_i\}$.

The results are incremental stresses and forces for the membrane and bending properties defined as follows

$$\left.\begin{aligned} \sigma_{x1} &= \sigma_x^* + \Delta\sigma_x \\ \sigma_{y1} &= \sigma_y^* + \Delta\sigma_y \\ \sigma_{xy1} &= \sigma_{xy}^* + \Delta\sigma_{xy} \end{aligned}\right\} \text{ Membrane stresses ,} \qquad (7), (8), (9)$$

$$\left.\begin{aligned} M_{x1} &= M_x^* + \Delta M_x \\ M_{x1} &= M_y^* + \Delta M_y \\ M_{xy1} &= M_{xy}^* + \Delta M_{xy} \\ V_{x1} &= V_x^* + \Delta V_x \\ V_{y1} &= V_y^* + \Delta V_y \end{aligned}\right\} \text{ Bending forces .} \qquad (10), (11), (12), (13), (14)$$

The total bending stresses are calculated using the total bending forces given in Equations 10 through 14 in conjunction with Equations 98 through 100 of section 4.87.5.7.

The membrane and bending stresses are added together as follows for Z_1:

$$\begin{Bmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{Bmatrix}_{\text{composite}} = \begin{Bmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{Bmatrix}_{\text{membrane}} + \begin{Bmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{Bmatrix}_{\text{bending}} ; \qquad (15)$$

and for Z_2:

$$\begin{Bmatrix} \sigma_{x2} \\ \sigma_{y2} \\ \sigma_{xy2} \end{Bmatrix}_{composite} = \begin{Bmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{Bmatrix}_{membrane} + \begin{Bmatrix} \sigma_{x2} \\ \sigma_{y2} \\ \sigma_{xy2} \end{Bmatrix}_{bending} . \quad (16)$$

The principal stresses and their orientation for output are calculated from the above results, as in Equations 28 through 30 of section 4.87.4.6.

In module PLA4, the bending properties are disregarded, and the ECPTNL data are rearranged to correspond tc the ECPT data for the TRMEM or QDMEM, and the stresses are calculated exactly as the ones for the TRMEM or QDMEM elements (see section 4.87.4.14).

In modules PLA3 and PLA4, after the above stress calculations have been completed, the next elastic coefficients are calculated in the same manner as Equations 106 through 112 of section 4.87.4.14.

The new ESTNL and ECPTNL entries are:

$$\varepsilon_0^* = \varepsilon^* , \quad (17)$$

$$\varepsilon^* = \varepsilon_1 , \quad (18)$$

$$E^* = E_1 , \quad (19)$$

$$\sigma_x^* = \sigma_{x1} , \quad (20)$$

$$\sigma_y^* = \sigma_{y1} , \quad (21)$$

$$\sigma_{xy}^* = \sigma_{xy1} , \quad (22)$$

$$M_x^* = M_{x1} , \quad (23)$$

$$M_y^* = M_{y1} , \quad (24)$$

$$M_{xy} = M_{xy1} , \quad (25)$$

$$V^*_{x1} = V_{x1} \quad , \tag{26}$$

$$V^*_{y1} = V_{y1} \quad . \tag{27}$$

In module PLA4, the ECPTNL data are rearranged, and the element stiffness matrices are calculated in the same manner as for the TRMEM or QDMEM elements in section 4.87.4.14.

4.87.7 The ELASi, MASSi and DAMPi Elements

The scalar elements (ELASi, MASSi and DAMPi, i = 1,2,3,4) are connected to scalar components of grid points or to scalar points. The ELASi elements contribute only to: a) the stiffness matrix, $[K^{x}_{gg}]$, for i = 1,2,3,4; and b) to the structural damping matrix, $[K^{4}_{gg}]$, for i = 1,2,3. The MASSi elements contribute only to the mass matrix, $[M_{gg}]$, and the DAMPi elements contribute only to the viscous damping matrix, $[B_{gg}]$.

The scalar elements do not require material or geometric properties in their calculations. Only the ELASi elements are used for stress or force calculations.

4.87.7.1 Input Data for the ELASi, MASSi and DAMPi Elements

The ECPT/EST entries for the scalar elements are:

Symbol	Description	Elements
SIL_1, SIL_2	Scalar indices of connected grid or scalar points	All
c_1, c_2	Component numbers corresponding to SIL_1 and SIL_2.	Types 1 and 2
K	Spring constant	All ELASi elements
g_e	Damping factor	ELAS1, 2 and 3
S	Stress coefficient	
B	Viscous damping coefficient	All DAMPi elements
m	Mass coefficient	All MASSi elements

4.87.7.2 ELASi Stiffness Matrix Generation (Subroutine KELAS of Module SMA1)

1. The two connected scalar indices are i_1 and i_2 given by:

$$i_1 = \begin{cases} SIL_1 + (c_1 - 1), \text{ if Point 1 is a grid point} \\ SIL_1, \text{ if Point 1 is a scalar point} \end{cases}$$

$$i_2 = \begin{cases} SIL_2 + (c_2 - 1), & \text{if Point 2 is a grid point} \\ SIL_2, & \text{if Point 2 is a scalar point} \end{cases}$$

2. The following terms are added to the $[K^x_{gg}]$ matrix:

   +K in position (i_1, i_1) and in position (i_2, i_2),

   -K in position (i_2, i_1) and in position (i_1, i_2).

3. If point 2 is not defined, add +K to position (i_1, i_2).

4. The damping terms are:

$$K^4 = K g_e . \tag{1}$$

These are added to $[K^4_{gg}]$ in the same manner as the stiffness terms were added to $[K^x_{gg}]$.

4.87.7.3 MASSi Mass Matrix Generation (Subroutine MASSD of Module SMA2)

These elements are treated like the ELASi elements except the "m" term is added to the four positions in $[M_{gg}]$.

4.87.7.4 DMAPi Damping Matrix Generation (Subroutine MASSD of Module SMA2)

These elements are treated like the ELASi elements except the "B" term is added to the four positions in $[B_{gg}]$.

4.87.7.5 ELASi Stress and Force Recovery (Subroutines SELAS1 and SELAS2 of Module SDR2)

The element force is:

$$F = K(u_2 - u_1), \tag{2}$$

where u_1 and u_2 are the displacements at scalar index numbers i_1 and i_2.

The element stress is:

$$\sigma = SF. \qquad (3)$$

4.87.8 Concentrated Mass Elements CØNM1, CØNM2

Two types of grid point mass data are available. CØNM1 defines the mass matrix directly at the point, with the axes defined by a given local coordinate system. CØNM2 defines the same matrix for a body with a mass and three inertias with a center of gravity offset from the grid point.

4.87.8.1 ECPT Entries for the CØNM1 Mass Element

Symbol	Description
N_m	Local coordinate system number in which the mass matrix is defined.
N_g, X, Y, Z	Local coordinate system number and basic coordinates of the point.
m_{11}	Terms of mass matrix given in row form (out to the diagonal term).
m_{21}, m_{22}	
m_{31},m_{32},m_{33}	
m_{41},m_{42},m_{43},m_{44}	
m_{51},m_{52},m_{53},m_{54},m_{55}	
m_{61},m_{62},m_{63},m_{64},m_{65},m_{66}	

4.87.8.2 Mass Matrix Calculations for the CØNM1 Element (Subroutine MCØNMX of Module SMA2)

1. Using the symmetrical relationships, fill out the remainder of the 6x6 matrix, [m]:

$$m_{ij} = m_{ji} \cdot \tag{1}$$

2. Using the basic coordinates of the point and the local coordinate system definition, the 3x3 transformation matrices $[T_g]$ and $[T_m]$ are generated, and the mass matrix in global coordinates is:

$$[M] = \left[\begin{array}{c|c} T_g^T & 0 \\ \hline 0 & T_g^T \end{array}\right] \left[\begin{array}{c|c} T_m & 0 \\ \hline 0 & T_m \end{array}\right] [m] \left[\begin{array}{c|c} T_m^T & 0 \\ \hline 0 & T_m^T \end{array}\right] \left[\begin{array}{c|c} T_g & 0 \\ \hline 0 & T_g \end{array}\right], \qquad (2)$$

where $[T_g]$ is the transformation from global-to-basic coordinates at the point, and $[T_m]$ is the transformation from the coordinates defined by the mass local system to the basic coordinate system.

4.87.8.3 ECPT Entries for the CØNM2 Mass Element

Symbol	Description
N_m	Local coordinate system number in which the mass terms are defined.
N_g, X, Y, Z	Local coordinate system number and basic coordinates of the point.
$\bar{m}$	Concentrated mass
x,y,z	Offset of center of gravity in mass coordinate system
I_{11} I_{21}, I_{22} I_{31}, I_{32}, I_{33}	Inertias about the center of gravity given in row order out to the diagonal term

4.87.8.4 Mass Matrix Calculations for the CØNM2 Element (Subroutine MCØNMX of Module SMA2)

1. The transformation from the offset to the grid point is:

$$[D] = \left[\begin{array}{ccc|ccc} 1 & & & 0 & z & -y \\ & 1 & & -z & 0 & x \\ & & 1 & y & -x & 0 \\ \hline & & & 1 & & \\ & 0 & & & 1 & \\ & & & & & 1 \end{array}\right]. \qquad (3)$$

2. The mass matrix referenced to the offset point is:

$$[m_o] = \left[\begin{array}{ccc:ccc} \bar{m} & & & & & \\ & \bar{m} & & & 0 & \\ & & \bar{m} & & & \\ \hdashline & & & I_{11} & -I_{21} & -I_{31} \\ & 0 & & -I_{21} & I_{22} & -I_{32} \\ & & & -I_{31} & -I_{32} & I_{33} \end{array}\right] . \quad (4)$$

3. The mass matrix about the grid point along the given coordinates is:

$$[m] = [D]^T[m_o][D]. \quad (5)$$

The actual nonzero terms of matrix [m] are calculated directly from the equations:

$$m_{11} = m_{22} = m_{33} = \bar{m} \quad , \quad (6)$$

$$m_{15} = m_{51} = -m_{24} = -m_{42} = +z\bar{m} \quad , \quad (7)$$

$$m_{16} = m_{61} = -m_{34} = -m_{43} = -y\bar{m} \quad , \quad (8)$$

$$m_{26} = m_{62} = -m_{35} = -m_{53} = x\bar{m} \quad , \quad (9)$$

$$m_{44} = I_{11} + (y^2 + z^2)\bar{m} \quad , \quad (10)$$

$$m_{55} = I_{22} + (x^2 + z^2)\bar{m} \quad , \quad (11)$$

$$m_{66} = I_{33} + (x^2 + y^2)\bar{m} \quad , \quad (12)$$

$$m_{45} = m_{54} = -I_{21} - xy\bar{m} \quad , \quad (13)$$

$$m_{46} = m_{64} = -I_{31} - xz\bar{m} \quad , \tag{14}$$

$$m_{56} = m_{65} = -I_{32} - yz\bar{m} \quad . \tag{15}$$

4. The matrix [m] is transformed back to global coordinates using Equation 2.

4.87.9 The CØNEAX Element

4.87.9.1 Input Data for the CØNEAX Element

1. The ECPT/EST entries for the CØNEAX element are:

Symbol	Description
SIL_a, SIL_b	Scalar indices of the n^{th} harmonic of the connected rings
n	Harmonic index
r_a, r_b	Radii at points a and b
z_a, z_b	Projected distances along the axis
t_m	Membrane thickness
$MAT\ ID_m$	Membrane material identification number
I	Bending coefficient
$MAT\ ID_b$	Bending material identification number
t_s	Shear thickness
$MAT\ ID_s$	Shear material identification number
Z_1, Z_2	Outer fiber distances for stress calculations
ϕ_i, i = 1 to 14	Angles defining points around element
t_μ	Temperature for material properties
μ	Nonstructural mass

4.87.9.2 Stiffness Matrix Calculations (Subroutine KCØNE of Module SMA1).

1. The shell orientation is given by:

$$\ell = \sqrt{(r_b - r_a)^2 + (z_b - z_a)^2}\ , \tag{1}$$

$$\sin\psi = \frac{r_b - r_a}{\ell}\ , \tag{2}$$

$$\cos\psi = \frac{z_b - z_a}{\ell}\ . \tag{3}$$

2. The transformation matrix [E] from element coordinates to ring cylindrical coordinates is:

$$[E] = \begin{bmatrix} 0 & \sin\psi & \cos\psi & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin\psi \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cos\psi \end{bmatrix} . \quad (4)$$

3. The serial steps for the balance of the stiffness matrix computations unique to the axisymmetric conical shell element are explicitly described in the NASTRAN Theoretical Manual section 5.9.5.7 (Summary of Procedures).

4.87.9.3 Mass Matrix Computation (Subroutine MCØNE of Module SMA2)

$$[M_{aa}] = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \quad (5)$$

where

$$m = \pi\ell\left(\frac{r_b}{6} + \frac{r_a}{3}\right)(\rho t + \mu) . \quad (6)$$

4.87.9.4 Element Load Calculations (Subroutine CØNE of Module SSG1).

The Fourier coefficients of the temperatures are stored in the GPTT data block. The loads are generated by the elements, which reference the connected rings and harmonics indirectly by the grid point scalar indices. The scalar indices are used with the SIL (Scalar Index List) data block to obtain the temperatures. The following steps are used to generate the loads:

1. The data for a logical element are read from the EST data block. The harmonic

number, n, is extracted from the element ID, N_e, by the equation:

$$N_e = 1000N + (n + 1) \quad , \tag{7}$$

where N is the total number of harmonics plus one in the problem.

The temperatures for this particular element and harmonic (T_n^a and T_n^b) are extracted from the GPTT data block. (No default nonzero temperatures are allowed).

2. The following data are generated in the same manner as in the stiffness matrix routine, KCØNE:

r_a, r_b, z_a, z_b	-	Ring locations
ℓ	-	Linear distance
$\sin\psi$, $\cos\psi$	-	Inclination functions
[E]	-	Element-to-global transformation matrix (6x5)
t	-	Element thickness
$[E_m]$	-	Material matrix (3x3)
$\alpha_s = \alpha_\phi = \alpha$	-	Temperature coefficient

3. The geometry coefficients, I_{mn}, are calculated as in the stiffness matrix routine by the equation:

$$I_{mn} = \pi \int_0^{\ell} s^m r^{1-n} \, ds = \frac{\pi}{b} \int_{r_a}^{r_b} s^m r^{1-n} \, dr \quad , \tag{8}$$

where $r = a + bs$, $a = r_a$, $b = \frac{r_b - r_a}{\ell}$.

4. The loads are generated in generalized coordinates, $\{P_n^q\}$, with the equations:

$$\{P_{1n}\} = n(I_{01}A_n + I_{11}B_n) \quad , \tag{9}$$

$$\{P_{2n}\} = n(I_{11}A_n + I_{21}B_n) \quad , \tag{10}$$

$$\{P_{3n}\} = \sin\psi\ (I_{01}A_n + I_{11}B_n) \ , \tag{11}$$

$$\{P_{4n}\} = \sin\psi(I_{11}A_n + I_{21}B_n) + I_{00}C_n + I_{10}D_n \ , \tag{12}$$

$$\{P_{5n}\} = \cos\psi(I_{01}A_n + I_{11}B_n) \ , \tag{13}$$

$$\{P_{6n}\} = \cos\psi(I_{11}A_n + I_{21}B_n) \ , \tag{14}$$

$$\{P_{7n}\} = \cos\psi(I_{21}A_n + I_{31}B_n) \ , \tag{15}$$

$$\{P_{8n}\} = \cos\psi(I_{31}A_n + I_{41}B_n) \ , \tag{16}$$

$$\{P_{9n}\} = 0 \ , \tag{17}$$

$$\{P_{10,n}\} = 0 \quad , \tag{18}$$

where

$$A_n = tE_{12}(\alpha_s T_n^a) + tE_{22}(\alpha_\phi T_n^a) \ , \tag{19}$$

$$B_n = \frac{t}{\ell}\,(T_n^b - T_n^a)(E_{12}\alpha_s + E_{22}\alpha_\phi), \tag{20}$$

$$C_n = tE_{11}(\alpha_s T_n^a) + tE_{12}(\alpha_\phi T_n^a) \ , \tag{21}$$

$$D_n = \frac{t}{\ell}\,(T_n^b - T_n^a)(E_{11}\alpha_s + E_{12}\alpha_\phi). \tag{22}$$

5. The transformation from generalized coordinates to element coordinates, $[G_{qu}]$, is calculated if no transverse shear material or thickness is given (i.e., MAT ID_s = 0 or t_s = 0).

$$[H_{qu}] = [H]^{-1}, \tag{23}$$

where [H] is given explicitly in the stiffness matrix calculations.

6. If transverse shear exists, the shear matrix, [B], is generated. Additional material terms are:

$[D] = I_b[E_b]$, where $[E_b]$ is the bending material 3x3 matrix.

$G_{11} = G$ = Shear coefficient of transverse shear material.

The nonzero terms of [B] are:

$$B_{4,1} = B_{9,1} = \frac{1}{Q}\left[D_{12}n\cos\psi\left(\frac{1}{r_b} - \frac{1}{r_a}\right) - \frac{n}{2}\cos\psi\sin\psi\,\frac{I_{03}}{\pi}(D_{33} + 2D_{22})\right], \tag{24}$$

$$B_{4,2} = B_{9,2} = \frac{1}{Q}\left[D_{12}\frac{n\ell\cos\psi}{r_b} - \frac{1}{2}n\sin\psi\cos\psi\,\frac{I_{13}}{\pi}(3D_{33} + D_{22}) + \frac{3}{2}nD_{33}\cos\psi\,\frac{I_{02}}{\pi}\right], \tag{25}$$

$$B_{4,3} = B_{9,3} = \frac{1}{Q}\left[\frac{1}{2}n^2D_{33}\cos\psi\,\frac{I_{03}}{\pi}\right], \tag{26}$$

$$B_{4,4} = B_{9,4} = \frac{1}{Q}\left[\frac{1}{2}n^2D_{33}\cos\psi\,\frac{I_{13}}{\pi}\right], \tag{27}$$

$$B_{4,5} = B_{9,5} = \frac{1}{Q}\left[n^2D_{12}\left(\frac{1}{r_b} - \frac{1}{r_a}\right) - n^2\sin\psi\,\frac{I_{03}}{\pi}(2D_{33} + D_{22})\right], \tag{28}$$

$$B_{4,6} + B_{9,6} = \frac{1}{Q}\left[D_{12}\frac{n^2\ell}{r_b} - n^2\sin\psi\,\frac{I_{13}}{\pi}(2D_{33} + D_{22}) + \frac{I_{02}}{\pi}(2n^2D_{33} + \sin^2\psi D_{22})\right], \tag{29}$$

$$B_{4,7} = B_{9,7} = \frac{1}{Q}\left[2D_{11}(r_a - r_b) + D_{12}\frac{n^2\ell^2}{r_b} + \frac{2I_{12}}{\pi}(2n^2D_{33} + \sin^2\psi\, D_{22})\right. \tag{30}$$

$$\left. - n^2\sin\psi\,\frac{I_{23}}{\pi}(2D_{33} + D_{22})\right],$$

$$B_{4,8} = B_{9,8} = \frac{1}{Q}\left[-D_{11}6\ell r_b + D_{12}\frac{n^2\ell^3}{r_b} + \frac{3I_{22}}{\pi}(2n^2D_{33} + \sin^2\psi\, D_{22})\right. \tag{31}$$

$$\left. - n^2\sin\psi\,\frac{I_{33}}{\pi}(2D_{33} + D_{22})\right],$$

$$B_{4,9} = B_{9,9} = \frac{1}{Q}\left[-n\,\sin\psi\,\frac{I_{02}}{\pi}(D_{22} + D_{33})\right], \tag{32}$$

$$B_{4,10} = B_{9,10} = \frac{1}{Q}\left[n\ell(D_{12} + D_{33}) - n\,\sin\psi\,\frac{I_{12}}{\pi}(D_{22} + D_{33})\right], \tag{33}$$

where

$$Q = \ell t_s G_{11}\frac{r_a + r_b}{2}\left[1 + \frac{I_{02}}{\pi}\frac{n^2D_{33} + \sin^2\psi\, D_{22}}{\ell t_s G_{11} r_{ar}}\right]. \tag{34}$$

7. The transformation $[\bar{H}]$ transforming displacements in the element coordinate system to displacements in generalized coordinates of the power series is:

$$[\bar{H}] = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\frac{\cos\psi}{r_a} & 0 & 0 & 0 & \frac{n}{r_a} & 0 & 0 & 0 & 1 & 0 \\
1 & \ell & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & \ell & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \ell & \ell^2 & \ell^3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 2\ell & 3\ell^3 & -1 & 0 \\
\frac{\cos\psi}{r_b} & \frac{\ell\cos\psi}{r_b} & 0 & 0 & \frac{n}{r_b} & \frac{n\ell}{r_b} & \frac{n\ell^2}{r_b} & \frac{n\ell^3}{r_b} & 1 & \ell
\end{bmatrix} \tag{35}$$

8. The transformation $[H_{qu}]$ for nonzero transverse shear is:

$$[H_{qu}] = ([\overline{H}] - [B])^{-1} \quad (10x10) \quad . \tag{36}$$

9. $[H_{qu}]$ is partitioned into two 10x5 matrices

$$[H_{qu}] = [H_a \mid H_b] \quad . \tag{37}$$

10. The loads in global coordinates are calculated with:

$$\{P_a\} = [E][H_a]^T\{P_n^q\} \quad , \tag{38}$$

$$\{P_b\} = [E][H_b]^T\{P_n^q\} \quad . \tag{39}$$

4.87.9.5 Element Stress Calculations (Subroutines SCØNE1, SCØNE2, SCØNE3 of Module SDR2)

1. For each element the following quantities are calculated as in section 4.87.9.2:

ℓ, $\sin\psi$, $\cos\psi$, $[E]$, $[H_a]$, $[H_b]$, $[H_{\varepsilon q}]$, $[H_{\psi q}]$, $[H_{\chi q}]$ (using $s = \frac{\ell}{2}$, $r = \frac{r_a + r_b}{2}$).

2. Using the material properties, the following matrices are calculated:

$$[E_m] = \frac{E_1}{(1 - \nu_1^2)} \begin{bmatrix} 1 & -\nu_1 & 0 \\ -\nu_1 & 1 & 0 \\ 0 & 0 & \frac{1 - \nu_1}{2} \end{bmatrix} \quad , \tag{40}$$

$$[D_b] = \frac{E_2 I}{(1 - \nu_2^2)} \begin{bmatrix} 1 & -\nu_2 & 0 \\ -\nu_2 & 1 & 0 \\ 0 & 0 & \frac{1 - \nu_2}{2} \end{bmatrix} , \quad (41)$$

$$[G_s] = \begin{bmatrix} G_3 & 0 \\ 0 & G_3 \end{bmatrix} , \quad (42)$$

where $[E_m]$, $[D_b]$ and $[G_s]$ are computed for membrane, bending and shear materials respectively.

3. The stress matrices are then calculated:

$$[K_s] = \begin{bmatrix} t[E_m]\ [H_{\varepsilon q}] \\ \text{- - - - - -} \\ [D]\ [H_{\chi q}] \\ \text{- - - - - -} \\ t_s[G_s]\ [H_{\gamma q}] \end{bmatrix} , \quad (8x10) \quad (43)$$

$$[S_a] = [K_s][H_a][E] , (8x6) \quad (44)$$

$$[S_b] = [K_s][H_b][E] , (8x6) . \quad (45)$$

4. Each entry in the EST data block contains data pertaining to the n^{th} harmonic motion of the CØNEAX element. The elements in the EST are ordered by harmonic and CØNEAX I.D. number. All harmonic elements for each CØNEAX are grouped together.

   a. When the harmonic, n, of an element is zero, this indicates it is the first of a group of elements. Storage space is allotted for fourteen 8 by 1 vectors defining the element forces at points. Two UGV vector data blocks must be used to calculate stresses on points. These data blocks correspond to the two subcases "C" and "S" and are solved simultaneously using the same data.

   b. Using the $[S_a]$ and $[S_b]$ matrices and the 6 by 1 displacement vectors, {u}, by their SIL numbers, stress and force vectors are computed.

If $n \neq 0$:

$$\begin{Bmatrix} \tilde{\sigma}^c_{sn} \\ \tilde{\sigma}^c_{\phi n} \\ \tilde{\sigma}^c_{s\phi n} \\ M^c_{sn} \\ M^c_{\phi n} \\ M^c_{s\phi n} \\ V^c_{sn} \\ V^c_{\phi n} \end{Bmatrix} = [S_{an}] \{u^c_{an}\} + [S_{bn}] \{u^c_{bn}\}, \tag{46}$$

$$\begin{Bmatrix} \tilde{\sigma}^s_{sn} \\ \tilde{\sigma}^s_{\phi n} \\ \tilde{\sigma}^s_{s\phi n} \\ M^s_{sn} \\ M^s_{\phi n} \\ M^s_{s\phi n} \\ V^s_{sn} \\ V^s_{\phi n} \end{Bmatrix} = [S_{an}] \{u^s_{an}\} + [S_{bn}] \{u^s_{bn}\}. \tag{47}$$

If n = 0:

$$\left\{\begin{array}{c} \bar{\sigma}_{so} \\ \bar{\sigma}_{\phi o} \\ \bar{\sigma}_{s\phi o} \\ M_{so} \\ M_{\phi o} \\ M_{s\phi o} \\ V_{so} \\ V_{\phi o} \end{array}\right\} = [S_{ao}] (\{u^s_{ao}\} + \{u^c_{ao}\}) - [S_{bo}] (\{u^s_{bo}\} + \{u^c_{bo}\}). \quad (48)$$

c. The temperature effects are added using the GPTT data for the two subcases, (T^c_α and T^s_α)

$$(\sigma^{c'}_{sn}) = \sigma^c_{sn} + \frac{1}{2} S_{tn} (T^c_a + T^c_b) , \quad (49)$$

$$(\sigma^{c'}_{\phi n}) = \sigma^c_{\phi n} + \frac{1}{2} S_{tn} (T^c_a + T^c_b) , \quad (50)$$

$$(\sigma^{c'}_{s\phi n}) = \sigma^c_{s\phi n} . \quad (51)$$

The same equations are used for the "S" set and when n = 0.

d. The harmonic stresses are calculated by the equations

for i = 1,2,3:

$$\sigma^c_{sni} = (\sigma^{c'}_{sn}) + \frac{M^c_{sn}\, ci}{I} , \quad (52)$$

$$\sigma^c_{\phi n} = (\sigma^{c'}_{\phi n}) + \frac{M^c_{\phi n}\, ci}{I} , \quad (53)$$

$$\sigma^c_{s\phi n} = (\sigma^{c'}_{s\phi n}) + \frac{M^c_{s\phi n}}{I} ci . \quad (54)$$

The equations are repeated for the S set and when n = 0.

e. Principal stresses (σ_1, σ_2, θ, τ_{max} etc.) are calculated as with the TRIA1 or QUAD1 element, except that when $n \neq 0$ the data are calculated for both the S and C sets.

f. The incremental element stresses or forces for the points on the cone are calculated from the following equations for j = 1,2,...,14:

For $n \neq 0$:

$$\delta\sigma_{sj} = (\sigma_{sn}^{C'}) \cos(n\phi_j) + (\sigma_{sn}^{S'}) \sin(n\phi_j) , \tag{55}$$

$$\delta\sigma_{\phi j} = (\sigma_{\phi n}^{C'}) \cos(n\phi_j) + (\sigma_{\phi n}^{S'}) \sin(n\phi_j) , \tag{56}$$

$$\delta\sigma_{s\phi j} = (\sigma_{s\phi n}^{C'}) \sin(n\phi_j) - (\sigma_{s\phi n}^{S'}) \cos(n\phi_j) , \tag{57}$$

$$\delta M_{sj} = M_{sn}^{C} \cos(n\phi_j) + M_{sn}^{S} \sin(n\phi_j), \tag{58}$$

$$\delta M_{\phi j} = M_{\phi n}^{C} \cos(n\phi_j) + M_{\phi n}^{S} \sin(n\phi_j) , \tag{59}$$

$$\delta M_{s\phi j} = M_{s\phi n}^{C} \sin(n\phi_j) - M_{s\phi n}^{S} \cos(n\phi_j) , \tag{60}$$

$$\delta V_{si} = V_{sn}^{C} \cos(n\phi_j) + V_{sn}^{S} \sin(n\phi_j), \tag{61}$$

$$\delta V_{\phi j} = V_{\phi n}^{C} \sin(n\phi_j) - V_{\phi n}^{S} \cos(n\phi_j) . \tag{62}$$

g. The incremental stress and force values are added to the running sums for the points. After the last element is calculated (n = N), the forces and stresses for the points are calculated and output. The equations are identical to steps d and e of this section except that 1) the "S" and "C" sets are not used and 2) up to 14 points may be calculated for output for each physical element.

Since the user may leave some spaces blank on the property card for this element, only one of the $\phi_j = 0$ points is used in the calculation.

4.87.9.6 Differential Stiffness Matrix Calculations (Subroutine DCØNE of Module DSMG1)

The data input from the ECPT to the DCØNE subroutine are the same as those given in section 4.87.9.1. Additional data for the generation of the differential stiffness matrix are as follows:

$\{u_a^o\}$, $\{u_b^o\}$ - Displacement vectors of the zero harmonic (extracted from the UGV data block)

T_a^o, T_b^o - Element loading temperatures of the zero harmonic (extracted from the GPTT data block)

The first part of the calculations involves calculation of the element force components. The following steps are performed with harmonic number n = 0.

1. The 10 by 10 transformation matrix $[H_{uq}^o]$ is computed from:

$$[H_{uq}^n] = [\bar{H}_{uq}] + \{H_{uY}\}\{H_{Y_s q}\}^T \tag{63}$$

where $[\bar{H}_{uq}]$, a 10 by 10 matrix, and $\{H_{uY}\}$, a column vector, are derived in the NASTRAN Theoretical Manual, section 5.9.5.3, and $\{H_{Y_s q}\}^T$, a row vector, is explicitly written out in Equation 85 of section 5.9 of the NASTRAN Theoretical Manual.

2. The 10 generalized displacement quantities q_i are

$$\{q\} = [H_{uq}^o]^{-1} \begin{Bmatrix} [E]^T \{u_a^o\} \\ \hline [E]^T \{u_b^o\} \end{Bmatrix} \tag{64}$$

where $\{u_a^o\}$ and $\{u_b^o\}$ are the 6 by 1 displacement vectors, and [E] is calculated as in Equation 4 of section 4.87.9.2.

3. The strain coefficients are

$$\begin{Bmatrix} \Delta\epsilon_s \\ \Delta\epsilon_\phi \\ \Delta\epsilon_{s\phi} \end{Bmatrix} = \{\alpha\} \frac{(T_b^c - T_a^o)}{\ell} \tag{65}$$

$$\begin{Bmatrix} \varepsilon_s \\ \varepsilon_\phi \\ \varepsilon_{s\phi} \end{Bmatrix} = T_a^o \{\alpha\} \tag{66}$$

where T_a^o and T_b^o are the loading temperatures at the grid points, the $\{\alpha\}$ vector is obtained from the MPT data block via subroutine MAT and ℓ is calculated as in Equation 1 of section 4.87.9.2.

4. The force coefficients are calculated:

$$a_o = t_m G_{12} (\sin \psi\, q_3 + \cos \psi\, q_5) \quad , \tag{67}$$

$$a_1 = t_m G_{12} (\sin \psi\, q_4 + \cos \psi\, q_6) \quad , \tag{68}$$

$$a_2 = t_m G_{12} \cos \psi\, q_7 \quad , \tag{69}$$

$$a_3 = t_m G_{12} \cos \psi\, q_8 \quad , \tag{70}$$

$$b_o = t_m G_{22} (\sin \psi\, q_3 + \cos \psi\, q_5) \quad , \tag{71}$$

$$b_1 = t_m G_{22} (\sin \psi\, q_4 + \cos \psi\, q_6) \quad , \tag{72}$$

$$b_2 = t_m G_{22} \cos \psi\, q_7 \quad , \tag{73}$$

$$b_3 = t_m G_{22} \cos \psi\, q_8 \quad , \tag{74}$$

$$c_o = t_m G_{11} (q_4 - \varepsilon_s) - t_m G_{12} \varepsilon_\phi \quad , \tag{75}$$

$$c_1 = -t_m G_{11} \Delta\varepsilon_s - t_m G_{12} \Delta\varepsilon_\phi \quad , \tag{76}$$

$$d_o = t_m G_{12} (q_4 - \varepsilon_s) - t_m G_{22} \varepsilon_\phi \quad , \tag{77}$$

$$d_1 = -t_m G_{12} \Delta\varepsilon_s - t_m G_{22} \Delta\varepsilon_\phi \quad , \tag{78}$$

where G_{11}, G_{12} and G_{22} are elements of the 3 by 3 symmetric material properties matrix, [G].

5. The geometry coefficients are calculated:

$$I_{mn} = \int_0^{\ell} s^m \, r^{1-n} \, ds \qquad \begin{cases} m = 0, 1, \ldots, 9 \\ n = 0, 1, 2, 3 \end{cases} ; \tag{79}$$

where

$$r = r_a + \left(\frac{r_b - r_a}{\ell} \right) s \quad . \tag{80}$$

An explicit formula for the evaluation of I_{mn} is given in the NASTRAN Theoretical Manual, section 5.9.5.8.

6. The following coefficients for the computation of the differential stiffness matrix are calculated:

$$A_{mn} = a_0 I_{m, n+1} + a_1 I_{m+1, n+1} + a_2 I_{m+2, n+1} + a_3 I_{m+3, n+1} + c_0 I_{m, n} + c_1 I_{m+1, n} \quad , \tag{81}$$

$$B_{mn} = b_0 I_{m, n+1} + b_1 I_{m+1, n+1} + b_2 I_{m+2, n+1} + b_3 I_{m+3, n+1} + d_0 I_{m, n} + d_1 I_{m+1, n} \quad , \tag{82}$$

$$C_{mn} = A_{mn} + B_{mn} \quad , \tag{83}$$

where m = 0, 1....6; n = 0, 1, 2.

Note: The index n used above is a dummy index and is not the harmonic number.

The second part of the calculations involves generating the differential stiffness matrix. The remaining steps use n as the harmonic number of the element.

1. The nonzero elements of the symmetric differential stiffness matrix $[K^{qd}]$, in generalized coordinates, are given in Table 1c below.

Table 1c. Nonzero Elements of the Differential Stiffness Matrix, $[K^{qd}]$.

$$K_{11}^{qd} = \cos^2\psi B_{02} + \frac{1}{4}\sin^2\psi C_{02}$$

$$K_{12}^{qd} = \cos^2\psi B_{12} + \frac{1}{4}\sin\psi C_{01} + \frac{1}{4}\sin^2\psi C_{12}$$

$$K_{13}^{qd} = \frac{1}{4}n\sin\psi C_{02}$$

$$K_{14}^{qd} = \frac{1}{4}n\sin\psi C_{12}$$

$$K_{15}^{qd} = n\cos\psi B_{02}$$

$$K_{16}^{qd} = n\cos\psi B_{12}$$

$$K_{17}^{qd} = n\cos\psi B_{22}$$

$$K_{18}^{qd} = n\cos\psi B_{32}$$

$$K_{22}^{qd} = \cos^2\psi B_{22} + \frac{1}{4}C_{00} + \frac{1}{2}\sin\psi C_{11} + \frac{1}{4}\sin^2\psi C_{22}$$

$$K_{23}^{qd} = \frac{1}{4}nC_{01} + \frac{1}{4}n\sin\psi C_{12}$$

$$K_{24}^{qd} = \frac{1}{4}nC_{11} + \frac{1}{4}n\sin\psi C_{22}$$

$$K_{25}^{qd} = n\cos\psi B_{12}$$

$$K_{26}^{qd} = n\cos\psi B_{22}$$

$$K_{27}^{qd} = n\cos\psi B_{32}$$

$$K_{28}^{qd} = n\cos\psi B_{42}$$

$$K_{33}^{qd} = \frac{1}{4}n^2C_{02}$$

$$K_{34}^{qd} = \frac{1}{4}n^2C_{12}$$

Table 1c (con'd). Elements of the Differential Stiffness Matrix, $[K^{qd}]$.

$$K^{qd}_{44} = \frac{1}{4} n^2 C_{22}$$

$$K^{qd}_{55} = n^2 B_{02}$$

$$K^{qd}_{56} = n^2 B_{12}$$

$$K^{qd}_{57} = n^2 B_{22}$$

$$K^{qd}_{58} = n^2 B_{32}$$

$$K^{qd}_{66} = A_{00} + n^2 B_{22}$$

$$K^{qd}_{67} = 2A_{10} + n^2 B_{32}$$

$$K^{qd}_{68} = 3A_{20} + n^2 B_{42}$$

$$K^{qd}_{77} = 4A_{20} + n^2 B_{42}$$

$$K^{qd}_{78} = 6A_{30} + n^2 B_{52}$$

$$K^{qd}_{88} = 9A_{40} + n^2 B_{62}$$

The formulas for n = 0 are the same as in Table 1c except that they are all multiplied by 2. The nonzero terms for n = 0 fall into two uncoupled sets which are

(11)	(12)
	(22)

Effect on Twisting

(66)	(67)	(68)
	(77)	(78)
		(88)

Effect on Axisymmetric Deformation

2. The 10 by 10 transformation matrix, $[H^n_{uq}]$, from generalized coordinates to element coordinates, for the n^{th} harmonic is computed as in Equation 63. The matrix is inverted and partitioned into two 10 by 5 matrices:

$$[H^n_{uq}]^{-1} \Longrightarrow [H_a \,|\, H_b] \quad . \tag{84}$$

3. The 6 by 6 differential stiffness matrices in global coordinates are:

$$[K^d_{ij}] = [E]\,[H_i]^T\,[K^{qd}]\,[H_j]\,[E]^T \quad , \tag{85}$$

where i = pivot grid point; j = a, b; and [E] is computed as in Equation 4 of section 4.87.9.2.

4.87.10 The TRIARG Element

4.87.10.1 Input Data for the TRIARG Element

1. The ECPT/EST entries for the axisymmetric triangular ring (TRIARG) element are:

Symbol	Descriptions
SIL_1, SIL_2, SIL_3	Scalar index numbers for the three grid points.
γ	Material property orientation angle (degrees)
Mat I.D.	Material property identification number.
N_1, X_1, Y_1, Z_1 N_2, X_2, Y_2, Z_2 N_3, X_3, Y_3, Z_3	Local coordinate system number and location in basic coordinates of the three grid points.
t_μ	Element temperature for material properties.

For this element, Y_i must equal zero for i = 1, 2 and 3 and, we define:

$$\{R_s\} = \begin{Bmatrix} X_1 \\ X_2 \\ X_3 \end{Bmatrix} , \tag{1}$$

$$\{Z_s\} = \begin{Bmatrix} Z_{s1} \\ Z_{s2} \\ Z_{s3} \end{Bmatrix} = \begin{Bmatrix} Z_1 \\ Z_2 \\ Z_3 \end{Bmatrix} . \tag{2}$$

2. Coordinate system data

The location (X_i, Y_i, Z_i) and local coordinate system number (N_i) of each grid point are used to calculate the 3 by 3 global-to-basic coordinate system transformation matrices, $[T_i]$, i = 1, 2, 3.

3. Material data

The material property identification number, Mat I.D., and the element temperature for material properties, t_μ, are used to select the following data items. For this element, material properties may be defined on a MAT1 or MAT3 but not a MAT2 bulk data card.

Symbol	Description
E_r, E_θ, E_z	Young's moduli in the radial, tangential and axial directions respectively.
$\nu_{r\theta}, \nu_{\theta z}, \nu_{zr}$	Poisson's ratios in the three directions indicated.
ρ	Mass density
$G_{r\theta}, G_{\theta z}, G_{rz}$	Shear moduli in the three directions indicated.
$\alpha_r, \alpha_\theta, \alpha_z$	Coefficients of thermal expansion in the three directions indicated.
T_o	Thermal expansion reference temperature.
g_e	Structural element damping coefficient.

4.87.10.2 General Geometric Calculations

1. Local coordinate calculations are:

$$Z_{min} = \text{minimum of } (Z_{s1}, Z_{s2}, Z_{s3}), \tag{3}$$

$$\{R_L\} = \{R_s\}, \tag{4}$$

$$\{Z_L\} = \{Z_s\} - \begin{Bmatrix} Z_{min} \\ Z_{min} \\ Z_{min} \end{Bmatrix}. \tag{5}$$

2. The transformation from field coordinates to grid point degrees of freedom is given by (R_{Li} and Z_{Li} are the i^{th} components of $\{R_L\}$ and $\{Z_L\}$ respectively):

$$[\overline{\Gamma}_{\beta q}] = \begin{bmatrix} 1 & R_{L1} & Z_{L1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & R_{L1} & Z_{L1} \\ 1 & R_{L2} & Z_{L2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & R_{L2} & Z_{L2} \\ 1 & R_{L3} & Z_{L3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & R_{L3} & Z_{L3} \end{bmatrix}_{6x6}, \tag{6}$$

$$[\Gamma_{\beta q}] = [\bar{\Gamma}_{\beta q}]^{-1} . \tag{7}$$

3. The transformation matrix from two to three degrees of freedom per point is:

$$[\Gamma_{qs}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{6x9} . \tag{8}$$

4.87.10.3 Integral Calculations

1. The integrals over the area of the cross-section are of the form:

$$\delta_{ij} = r^i z^j dz dr, \tag{9}$$

for the values:

$$\delta_{-10}, \delta_{-11}, \delta_{00}, \delta_{10}, \delta_{20}, \delta_{30}, \delta_{01}, \delta_{11}, \delta_{21}, \delta_{02}, \delta_{12}, \delta_{-12} .$$

To accomplish this we integrate in two parts:

a. From line $Z = K_{12}r + m_{12}$ to line $Z = K_{13}r + m_{13}$

where

$$K_{ij} = \frac{Z_{Lj} - Z_{Li}}{R_{Lj} - R_{Li}}, \tag{10}$$

and

$$m_{ij} = - \frac{R_{Li}Z_{Lj} - R_{Lj}Z_{Li}}{R_{Lj} - R_{Li}}, \tag{11}$$

and from point R_{L1} to point R_{L3}.

b. From line $Z = K_{12}r + m_{12}$ to $Z = K_{32}r + m_{32}$ and from point R_{L3} to R_{L2}.

For the case where

$$R_{L1} = R_{L2} \text{ or } \left|\frac{R_{L2} - R_{L1}}{R_{L2}}\right| < 10^{-5}$$

we must integrate differently, that is, from line $Z = K_{32}r + m_{32}$ to $Z = K_{13}r + m_{13}$ and from R_{L1} to R_{L3}.

2. After the integrals are computed, a check is made to determine if an excessive amount of round-off error occurred. If round-off was excessive, an approximate integral can be calculated.

These tests are:

If any $\delta_{ij}<0$, then approximation must be used.

If $\sigma_{12} \leq \sigma_{02}$, or $\delta_{-12} \geq \delta_{12}$, or $\delta_{-12} > \delta_{02}$, then approximation must be used.

If $\Delta r \leq \hat{r}$ or $\Delta Z \leq \hat{Z}$, then approximation must be used. The terms Δr, ΔZ, $\hat{r}$ and $\hat{Z}$ are:

$$\Delta r = \max.(|R_{L1} - R_{L2}|, |R_{L2} - R_{L3}|, |R_{L3} - R_{L1}|), \tag{12}$$

$$\hat{r} = [\min.(R_{L1}, R_{L2}, R_{L3})]/10, \tag{13}$$

$$\Delta Z = \max.(|Z_{L1} - Z_{L2}|, |Z_{L2} - Z_{L3}|, |Z_{L3} - Z_{L1}|), \tag{14}$$

$$\hat{Z} = [\min (Z_{L1}, Z_{L2}, Z_{L3})]/10. \tag{15}$$

The approximation is:

$$\sigma_{ij} = (r_a)^i (Z_a)^j A, \tag{16}$$

where

$$r_a = \frac{1}{3}[R_{L1} + R_{L2} + R_{L3}], \tag{17}$$

$$Z_a = \frac{1}{3}[Z_{L1} + Z_{L2} + Z_{L3}], \tag{18}$$

$$A = \frac{1}{2}[R_{L1}\,(Z_{L2}-Z_{L3}) + R_{L2}\,(Z_{L3}-Z_{L1}) + R_{L3}\,(Z_{L1}-Z_{L2})]. \tag{19}$$

3. Form the matrix of integrals:

$$[\tilde{D}] = 2\pi \begin{bmatrix} 0 & \delta_{10} & 0 & 0 & 0 & 0 \\ \delta_{00} & \delta_{10} & \delta_{01} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \delta_{10} \\ 0 & 0 & \delta_{10} & 0 & \delta_{10} & 0 \end{bmatrix}_{4x6}. \tag{20}$$

4.87.10.4 Elastic Constants Matrix Calculations

1. Generate the transformation from material axis to element geometric axis:

$$[T_{eo}] = \begin{bmatrix} \cos^2\gamma & 0 & \sin^2\gamma & \sin\gamma\cos\gamma \\ 0 & 1 & 0 & 0 \\ \sin^2\gamma & 0 & \cos^2\gamma & -\sin\gamma\cos\gamma \\ -2\sin\gamma\cos\gamma & 0 & 2\sin\gamma\cos\gamma & \cos^2\gamma-\sin^2\gamma \end{bmatrix}_{4x4}. \tag{21}$$

2. Generate the matrix of elastic constants for an orthotropic body with respect to cylindrical coordinates:

$$[E_m] = \frac{1}{\Delta} \begin{bmatrix} E_r(1-\nu_{\theta z}\nu_{z\theta}) & & & \\ E_r(\nu_{\theta r}+\nu_{zr}\nu_{\theta z}) & E_\theta(1-\nu_{rz}\nu_{zr}) & \text{(Symmetric)} & \\ E_r(\nu_{zr}+\nu_{\theta r}\nu_{z\theta}) & E_\theta(\nu_{z\theta}+\nu_{r\theta}\nu_{zr}) & E_z(1-\nu_{r\theta}\nu_{\theta r}) & \\ 0 & 0 & 0 & G_{rz}\Delta \end{bmatrix}_{4x4} , \quad (22)$$

where

$$\nu_{\theta r} = \nu_{r\theta} E_\theta/E_r , \quad (23)$$

$$\nu_{z\theta} = \nu_{\theta z} E_z/E_\theta , \quad (24)$$

$$\nu_{rz} = \nu_{zr} E_r/E_z , \quad (25)$$

$$\Delta = 1 - \nu_{r\theta}\nu_{\theta r} - \nu_{\theta z}\nu_{z\theta} - \nu_{zr}\nu_{rz} - \nu_{r\theta}\nu_{\theta z}\nu_{zr} - \nu_{rz}\nu_{\theta r}\nu_{z\theta} . \quad (26)$$

3. Calculate the elastic constants matrix in element geometric axes:

$$[E_g] = [T_{eo}]^T[E_m][T_{eo}] . \quad (27)$$

4.87.10.5 Stiffness Matrix Generation (Subroutine KTRIRG of Module SMA1)

1. Generate the element stiffness matrix in field coordinates:

$$[\tilde{K}] =$$

$$2\pi \begin{bmatrix} E_{22}\delta_{-10} & & & & & \\ (E_{12}+E_{22})\delta_{00} & (E_{11}+2E_{12}+E_{22})\delta_{10} & & & & \\ E_{22}\delta_{-11}+E_{24}\delta_{00} & (E_{12}+E_{22})\delta_{01}+(E_{14}+E_{24})\delta_{10} & E_{22}\delta_{-12}+2E_{24}\delta_{01}+E_{44}\delta_{10} & & & \\ 0 & 0 & 0 & 0 & & \\ E_{24}\delta_{00} & (E_{14}+E_{24})\delta_{10} & E_{24}\delta_{01}+E_{44}\delta_{10} & 0 & E_{44}\delta_{10} & \\ E_{23}\delta_{00} & (E_{13}+E_{23})\delta_{10} & E_{23}\delta_{01}+E_{34}\delta_{10} & 0 & E_{34}\delta_{10} & E_{33}\delta_{10} \end{bmatrix}_{6\times 6} \quad (28)$$

where E_{ij} is an element of $[E_g]$.

2. Transform the element stiffness matrix from field coordinates to grid point degrees of freedom:

$$[\bar{K}] = [\Gamma_{\beta q}]^T[\tilde{K}][\Gamma_{\beta q}]. \quad (29)$$

3. Transform the element stiffness matrix from two to three degrees of freedom per point.

$$[K] = [\Gamma_{qs}]^T[\bar{K}][\Gamma_{qs}]. \quad (30)$$

4. The 3 by 3 partitions $[K^3_{pj}]$ of [K] corresponding to the pivot point p are transformed:

$$[K^3_{pj}] = [T_p]^T[K^3_{pj}][T_j], \quad j = 1,2,3 \quad (31)$$

5. Finally these 3 by 3 partitions are expanded to 6 by 6 partitions:

$$[K_{pj}] = \begin{bmatrix} K^3_{pj} & 0 \\ 0 & 0 \end{bmatrix} \quad (32)$$

4.87.10.6 Mass Matrix Calculations (Subroutine MTRIRG of Module SMA2)

1. Generate the consistent mass matrix in field coordinates:

$$[M] = 2\pi \begin{bmatrix} \bar{m}_{11}\delta_{10} & & & & & \\ \bar{m}_{11}\delta_{20} & \bar{m}_{11}\delta_{30} & & & \text{(Symmetric)} & \\ \bar{m}_{11}\delta_{11} & \bar{m}_{11}\delta_{21} & \bar{m}_{11}\delta_{12} & & & \\ 0 & 0 & 0 & \bar{m}_{22}\delta_{10} & & \\ 0 & 0 & 0 & \bar{m}_{22}\delta_{20} & \bar{m}_{22}\delta_{30} & \\ 0 & 0 & 0 & \bar{m}_{22}\delta_{11} & \bar{m}_{22}\delta_{21} & \bar{m}_{22}\delta_{12} \end{bmatrix}_{6\times 6} \quad (33)$$

where

$$[\bar{m}] = \begin{bmatrix} \rho & 0 \\ 0 & \rho \end{bmatrix} = \begin{bmatrix} \bar{m}_{11} & 0 \\ 0 & \bar{m}_{22} \end{bmatrix}. \quad (34)$$

2. Transform the mass matrix from field coordinates to grid point degrees of freedom:

$$[\tilde{M}] = [\Gamma_{\beta q}]^T [\tilde{M}][\Gamma_{\beta q}]. \quad (35)$$

3. Transform the mass matrix from two to three degrees of freedom per point:

$$[M] = [\Gamma_{qs}]^T [\bar{M}][\Gamma_{qs}]. \quad (36)$$

4. The 6 by 6 partitions, $[M_{pj}]$, are calculated as in Equations 31 and 32.

4.87.10.7 Thermal Load Calculations (Subroutine TTRIRG of Module SSG1)

1. Form the vector of thermal strains:

$$\{\bar{\alpha}\} = T_{avg} \begin{Bmatrix} \alpha_r \\ \alpha_\theta \\ \alpha_z \\ 0 \end{Bmatrix} \quad (4x1). \tag{37}$$

where T_{avg} is the average loading temperature at the grid points.

2. Compute thermal load vector in grid point degrees of freedom:

$$\{\bar{F}_T\} = [\Gamma_{\beta q}]^T [\tilde{D}]^T [E_g]\{\alpha\} \quad (6x1). \tag{38}$$

3. Transform thermal load from two to three degrees of freedom per point

$$\{F_T\} = [\Gamma_{qs}]^T \{\bar{F}_T\} \quad (9x1). \tag{39}$$

4. Each partition, $\{F_T^3\}$, of length 3 of $\{F_T\}$ is transformed to global coordinates by

$$\{F_T^3\}_g = [T_i]^T \{F_T^3\} \ . \tag{40}$$

5. These vectors are added to the overall load vector, $\{P_g\}$.

4.87.10.8 Element Force and Stress Calculations (Subroutines STRIR1 and STRIR2 of Module SDR2)

Element stress and force data items are calculated in two phases. The first phase (subroutine STRIR1) calculates the element stiffness and stress matrices. The second phase (subroutine STRIR2) calculates the actual element forces and stresses from the various subcase displacement vectors.

Phase 1 calculations are as follows:

1. Form the element stiffness matrix, [K], as in section 4.87.10.5.

2. Compute the constants:

$$r_o = \frac{1}{3} \sum_{i=1}^{3} R_{Li}, \tag{41}$$

$$z_o = \frac{1}{3} \sum_{i=1}^{3} Z_{Li}. \tag{42}$$

3. Form the $[D_o]$ matrix:

$$[D_o] = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{r_o} & 1 & \frac{z_o}{r_o} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}_{4x6}. \tag{43}$$

4. Compute the stress matrix

$$[\bar{S}] = [E_g][D_o][\Gamma_{\beta q}]. \tag{44}$$

5. Transform the stress matrix from two to three degrees of freedom per point:

$$[S] = [\bar{S}][\Gamma_{qs}]. \tag{45}$$

6. Transform the stress matrix from basic to local coordinates:

$$[S]_\ell = [S][T_{123}], \tag{46}$$

where

$$[T_{123}] = \left[\begin{array}{c|c|c} T_1 & 0 & 0 \\ \hline 0 & T_2 & 0 \\ \hline 0 & 0 & T_3 \end{array}\right] . \tag{47}$$

7. Compute the thermal stress vector

$$\{T_s\} = [E_g]\{\bar{\alpha}\} . \tag{48}$$

Phase 2 calculations are as follows:

1. Extract the displacement vector, $\{\Delta\}$, at the three translational components of the grid points from the global displacement vector.

2. Calculate the element forces:

$$\{P\} = [K]\{\Delta\} . \tag{49}$$

3. Calculate the element stresses:

$$\{\sigma\} = [S]_\ell\{\Delta\} - T_d\{T_s\} , \tag{50}$$

where

$$T_d = T_{avg} - T_o . \tag{51}$$

4.87.11 The TRAPRG Element

4.87.11.1 Input Data for the TRAPRG Element

1. The ECPT/EST entries for the axisymmetric trapezoidal ring (TRAPRG) element are

Symbol	Description
$SIL_1, SIL_2, SIL_3, SIL_4$	Scalar index numbers for the four grid points.
γ	Material property orientation angle (degrees).
Mat. I.D.	Material property identification number.
N_1, X_1, Y_1, Z_1 N_2, X_2, Y_2, Z_2 N_3, X_3, Y_3, Z_3 N_4, X_4, Y_4, Z_4	Local coordinate system number and location in basic coordinates of the four grid points.
t_μ	Element temperature for material properties

For this element Y_i must equal zero for i = 1, 2, 3 and 4, and we define:

$$\{R_s\} = \begin{Bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{Bmatrix} , \tag{1}$$

$$Z_s = \begin{Bmatrix} Z_{s1} \\ Z_{s2} \\ Z_{s3} \\ Z_{s4} \end{Bmatrix} = \begin{Bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \end{Bmatrix} . \tag{2}$$

2. Geometry Input

The location (X_i, Y_i, Z_i) and local coordinate system number (N_i) of each grid point are used to calculate the 3 by 3 global-to-basic coordinate system transformation matrices, $[T_i]$, i = 1,2,3,4.

3. Material Property Input

The material property input for the TRAPRG element is the same as that for the TRIARG element (see section 4.87.10.1).

4.87.11.2 General Calculations

1. Local coordinate calculations are:

$$Z_{min} = \text{minimum of } (Z_{s1}, Z_{s2}, Z_{s3}, Z_{s4}), \tag{3}$$

$$\{R_L\} = \{R_s\}, \tag{4}$$

$$\{Z_L\} = \{Z_s\} - \begin{Bmatrix} Z_{min} \\ Z_{min} \\ Z_{min} \\ Z_{min} \end{Bmatrix}. \tag{5}$$

2. Test for a rectangle. Let R_{Li} be the i^{th} component of $\{R_L\}$, and define:

$$R_{M14} = (R_{L1} + R_{L4})/2, \tag{6}$$

$$R_{M23} = (R_{L2} + R_{L3})/2. \tag{7}$$

If $\left|\frac{R_{L1} - R_{L4}}{R_{M14}}\right| < 0.005$ then $R_{L1} = R_{L4} = R_{M14}$.

If $\left|\frac{R_{L2} - R_{L3}}{R_{M23}}\right| < 0.005$ then $R_{L2} = R_{L3} = R_{M23}$.

If $R_{L1} = R_{L4}$ and $R_{L2} = R_{L3}$, then the element is rectangular.

If $R_{L1} = R_{L4} = 0$, then the element is a core element.

3. Generate the transformation matrix from field coordinates to grid point degrees of freedom:

$$[\bar{H}] = \begin{bmatrix} 1 & R_{L1} & Z_{L1} & R_{L1}Z_{L1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L1} & Z_{L1} & R_{L1}Z_{L1} \\ 1 & R_{L2} & Z_{L1} & R_{L2}Z_{L1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L2} & R_{L1} & R_{L2}Z_{L1} \\ 1 & R_{L3} & Z_{L4} & R_{L3}Z_{L4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L3} & Z_{L4} & R_{L3}Z_{L4} \\ 1 & R_{L4} & Z_{L4} & R_{L4}Z_{L4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L4} & Z_{L4} & R_{L4}Z_{L4} \end{bmatrix}_{8x8} . \quad (8)$$

$$[H] = [\bar{H}]^{-1} . \quad (9)$$

4. Generate the transformation matrix from two to three degrees of freedom per point:

$$[\Gamma_{qs}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{8x12} . \quad (10)$$

5. If the element is a core element, then:

$$\begin{aligned} h_{1j} &= h_{3j} = 0 \qquad j = 1,2,\ldots,8 , \\ h_{i1} &= h_{i7} = 0 \qquad i = 1,2,\ldots,8 . \end{aligned} \quad (11)$$

where h_{ij} is an element of [H].

4.87.11.3 Integral Calculations

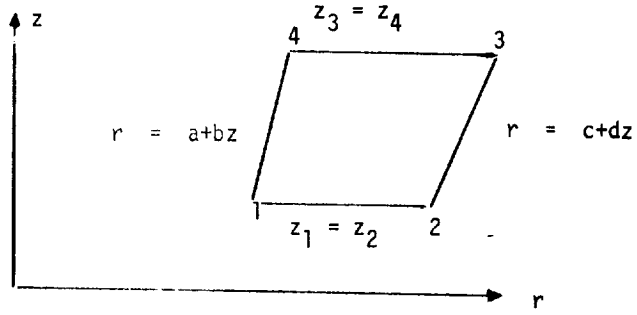
a. Compute the integrals over the cross-section of the trapezoid that are of the form:

$$I_{pq} = \iint_{rz} r^p z^q \, dr dz \tag{12}$$

for the values:

$I_{-10}, I_{-11}, I_{-12}, I_{00}, I_{01}, I_{02}, I_{10}, I_{11}, I_{12}, I_{20}, I_{21}, I_{22}, I_{30}, I_{31}, I_{32}$.

The limits of integration are chosen depending on the geometric shape of the trapezoid.



If the lines between points 1 and 4 and between points 2 and 3 are defined by the equations

$$r = a + bz, \tag{13}$$

$$r = c + dz, \tag{14}$$

respectively, then:

$$a = R_{L1} - \left(\frac{R_{L4} - R_{L1}}{Z_{L4} - Z_{L1}}\right) Z_{L1}, \tag{15}$$

$$b = \frac{R_{L4} - R_{L1}}{Z_{L4} - Z_{L1}}, \tag{16}$$

$$c = R_{L2} - \left(\frac{R_{L3} - R_{L2}}{Z_{L3} - Z_{L2}}\right) Z_{L2}, \tag{17}$$

$$d = \frac{R_{L3} - R_{L2}}{Z_{L3} - Z_{L2}}. \tag{18}$$

In general, the integration takes the form:

$$I_{Pq} = \int_{z_1}^{z_4} \int_{a+bz}^{c+dz} r^P z^q \, drdz. \tag{19}$$

For the case with the side r = c + dz parallel to the axis of symmetry (the z axis) we have:

$$I_{Pq} = \int_{z_1}^{z_4} \int_{a+bz}^{c} r^P z^q \, drdz. \tag{20}$$

For the case with the side r = a + bz parallel to the axis of symmetry we have:

$$I_{Pq} = \int_{z_1}^{z_4} \int_{a}^{c+dz} r^P z^q \, drdz. \tag{21}$$

And finally for the rectangle, the integration takes the form

$$I_{Pq} = \int_{z_1}^{z_4} \int_{a}^{c} r^P z^q \, dr dz. \tag{22}$$

4.87.11.4 Elastic Constants Matrix Calculation

The elastic constants matrix in element coordinates, $[E_g]$, for the TRAPRG element is calculated identically to the elastic constants matrix for the TRIARG element (see section 4.87.10.4).

4.87.11.5 Stiffness Matrix Generation (Subroutine KTRAPR of Module SMA1)

1. Generate the terms of the symmetric element stiffness matrix in field coordinates as shown in Table 2. Each term must be multiplied by 2π to form $[\tilde{K}]$.

2. Transform the element stiffness matrix from field coordinates to grid point degrees of freedom:

$$[\bar{K}] = [H]^T[\tilde{K}][H]. \tag{23}$$

3. Transform the element stiffness matrix from two to three degrees of freedom per point:

$$[K] = [\Gamma_{qs}]^T[\bar{K}][\Gamma_{qs}]. \tag{24}$$

4. The 3 by 3 partitions $[K_{pj}]$ of $[K]$ corresponding to the pivot point p are transformed:

$$[K^3_{pj}] = [T_p]^T[K^3_{pj}][T_j], \quad j = 1,2,3,4. \tag{25}$$

5. Finally, these 3 by 3 partitions are expanded to 6 by 6 partitions:

$$[K_{pj}] = \left[\begin{array}{c|c} K^3_{pj} & 0 \\ \hline 0 & 0 \end{array}\right]. \tag{26}$$

Table 2. Elements of the 8 by 8 Symmetric Stiffness Matrix for the TRAPRG Element.

	Col. 1	Col. 2	Col. 3	Col. 4
Row 1	$E_{22}I_{-10}$			
Row 2	$(E_{22}+E_{12})I_{00}$	$(E_{11}+2E_{12}+E_{22})I_{10}$		
Row 3	$E_{22}I_{-11}$	$(E_{12}+E_{22})I_{01}$	$E_{22}I_{-12}+E_{44}I_{10}$	
Row 4	$(E_{12}+E_{22})I_{01}$	$(E_{11}+2E_{12}+E_{22})I_{11}$	$(E_{12}+E_{22})I_{02}$ + $E_{44}I_{20}$	$(E_{11}+2E_{12}+E_{22})I_{12}$ + $E_{44}I_{30}$
Row 5	0	0	0	0
Row 6	0	0	$E_{44}I_{10}$	$E_{44}I_{20}$
Row 7	$E_{32}I_{00}$	$(E_{13}+E_{23})I_{10}$	$E_{23}I_{01}$	$(E_{13}+E_{23})I_{11}$
Row 8	$E_{32}I_{10}$	$(E_{31}+E_{32})I_{20}$	$(E_{23}+E_{44})I_{11}$	$(E_{13}+E_{23}+E_{44})I_{21}$

	Col. 5	Col. 6	Col. 7	Col. 8
Row 5	0			
Row 6	0	$E_{44}I_{10}$		
Row 7	0	0	$E_{33}I_{10}$	
Row 8	0	$E_{44}I_{11}$	$E_{33}I_{20}$	$E_{33}I_{30}+E_{44}I_{12}$

4.87.11.6 Mass Matrix Calculation (Subroutine MTRAPR of Module SMA2)

1. Form the coupled mass matrix in field coordinates:

$$[\tilde{M}] = 2\pi \begin{bmatrix} M_1 I_{10} & & & & & & & \\ M_1 I_{20} & M_1 I_{30} & & & \text{Symmetric} & & & \\ M_1 I_{11} & M_1 I_{21} & M_1 I_{12} & & & & & \\ M_1 I_{21} & M_1 I_{31} & M_1 I_{22} & M_1 I_{32} & & & & \\ 0 & 0 & 0 & 0 & M_2 I_{10} & & & \\ 0 & 0 & 0 & 0 & M_2 I_{20} & M_2 I_{30} & & \\ 0 & 0 & 0 & 0 & M_2 I_{11} & M_2 I_{21} & M_2 I_{12} & \\ 0 & 0 & 0 & 0 & M_2 I_{21} & M_2 I_{31} & M_2 I_{22} & M_2 I_{32} \end{bmatrix}_{8x8} \cdot \tag{27}$$

where

$$M_1 = M_2 = \rho \cdot \tag{28}$$

2. Transform the mass matrix to grid point degrees of freedom:

$$[\bar{M}] = [H]^T[\tilde{M}][H]. \tag{29}$$

3. Transform the mass matrix from two to three degrees of freedom per point:

$$[M] = [\Gamma_{qs}][\bar{M}][\Gamma_{qs}]. \tag{30}$$

4. The 6 by 6 partitions, $[M_{pj}]$, are calculated as in Equations 25 and 26.

4.87.11.7 Thermal Load Calculations (Subroutine TTRAPR of Module SSG1)

1. Form the temperature gradient vector:

$$\{\Delta T\} = \{T\} - \begin{Bmatrix} T_o \\ T_o \\ T_o \\ T_o \end{Bmatrix} , \tag{31}$$

where $\{T\}$ is the vector of loading temperatures at the grid points.

2. Form the thermal expansion coefficient vector:

$$\{\alpha\} = \begin{Bmatrix} \alpha_r \\ \alpha_\theta \\ \alpha_z \\ 0 \end{Bmatrix} . \tag{32}$$

3. Multiply the elastic constants matrix by the thermal expansion coefficient vector:

$$\{AB\} = [E_g]\{\alpha\} . \tag{33}$$

4. Form the [Q] matrix:

$$[Q] = \begin{bmatrix} AB_2I_{00} & AB_2I_{10} & AB_2I_{01} & AB_2I_{11} \\ (AB_1+AB_2)I_{10} & (AB_1+AB_2)I_{20} & (AB_1+AB_2)I_{11} & (AB_1+AB_2)I_{21} \\ AB_2I_{01} & AB_2I_{11} & AB_2I_{02} & AB_2I_{12} \\ (AB_1+AB_2)I_{11} & (AB_1+AB_2)I_{21} & (AB_1+AB_2)I_{12} & (AB_1+AB_2)I_{22} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ AB_3I_{10} & AB_3I_{20} & AB_3I_{11} & AB_3I_{21} \\ AB_3I_{20} & AB_3I_{30} & AB_3I_{21} & AB_3I_{31} \end{bmatrix}_{8x4} . \tag{34}$$

5. Partition the transformation matrix [H] to form [H']:

$$[H'] = [h'_{ij}] \qquad 4x4, \tag{35}$$

where

$$h'_{ij} = h_{ik} \quad \text{for } i = 1,2,4; \quad j = 1,2,\ldots,4; \text{ and } k = 2j-1$$

and h_{ik} are the elements of [H].

6. Compute the thermal load in field coordinates:

$$\{\tilde{F}_T\} = 2\pi[Q][H']\{\Delta T\}. \tag{36}$$

7. Transform the thermal load to grid point degrees of freedom:

$$\{\bar{F}_T\} = [H]^T\{\tilde{F}_T\}. \tag{37}$$

8. Transform the thermal load from two to three degrees of freedom per point:

$$\{F_T\} = [\Gamma_{qs}]^T\{\bar{F}_T\}. \tag{38}$$

9. Each partition, $\{F_T^3\}$, of length 3 of $\{F_T\}$ is transformed to global coordinates by:

$$\{F_T^3\}_g = [T_i]^T\{F_T^3\}. \tag{39}$$

10. These vectors are added to the overall load vector, $\{P_g\}$.

4.87.11.8 Element Force and Stress Calculations (Subroutines STRAP1 and STRAP2 of Module SDR2).

Element stress and force data items are calculated in two phases. The first phase (subroutine STRAP1) calculates the element stiffness and stress matrices. The second phase (subroutine STRAP2) calculates the element forces and stresses from the various subcase displacement vectors.

Phase 1 calculations are as follows:

1. Form the element stiffness matrix, [K], as in section 4.87.11.5.

2. Define a fifth "grid point" to be the average of the other four points:

$$R_{L5} = \frac{1}{4}(R_{L1} + R_{L2} + R_{L3} + R_{L4}), \tag{40}$$

$$Z_{L5} = \frac{1}{4}(Z_{L1} + Z_{L2} + Z_{L3} + Z_{L4}). \tag{41}$$

3. Form the matrices $[D^{(1)}],[D^{(2)}],[D^{(3)}],[D^{(4)}],[D^{(5)}]$

where

$$[D^{(i)}] = \begin{bmatrix} 0 & 1 & 0 & Z_{Li} & 0 & 0 & 0 & 0 \\ \frac{1}{R_{Li}} & 1 & \frac{Z_{Li}}{R_{Li}} & Z_{Li} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_{Li} \\ 0 & 0 & 1 & R_{Li} & 0 & 1 & 0 & Z_{Li} \end{bmatrix}_{4x8}, \tag{42}$$

where i = 1 to 5 denotes the five grid points.

4. Compute the stress matrices for each of the five grid points in field coordinates:

$$[\tilde{S}^{(i)}] = [E_g][D^{(i)}]. \tag{43}$$

5. Transform each stress matrix to grid point degrees of freedom:

$$[\bar{S}^{(i)}] = [\tilde{S}^{(i)}][H]. \tag{44}$$

6. Transform each stress matrix from two to three degrees of freedom per point:

$$[S^{(i)}] = [\bar{S}^{(i)}][\Gamma_{qs}]. \tag{45}$$

7. Form the master stress matrix:

$$[S] = \begin{bmatrix} S^{(1)} \\ S^{(2)} \\ S^{(3)} \\ S^{(4)} \\ S^{(5)} \end{bmatrix} \cdot \quad 20 \times 12 \tag{46}$$

8. Transform the stress matrix from basic to local coordinates:

$$[S_\ell] = [S][T_{1234}], \tag{47}$$

where

$$[T_{1234}] = \left[\begin{array}{c|c|c|c} T_1 & 0 & 0 & 0 \\ \hline 0 & T_2 & 0 & 0 \\ \hline 0 & 0 & T_3 & 0 \\ \hline 0 & 0 & 0 & T_4 \end{array}\right] \cdot \tag{48}$$

9. Compute the thermal stress vector:

$$\{T_s\} = [E_g]\{\alpha\} \cdot \tag{49}$$

Phase 2 calculations are as follows:

1. Extract the displacement vector, $\{\Delta\}$, at the three translation coordinates of each of the four grid points from the global displacement vector.

2. Calculate the element forces:

$$\{P\} = [K]\{\Delta\} \cdot \tag{50}$$

3. Calculate the element stresses:

$$\{\sigma\} = [S_\ell]\{\Delta\} - T_d \{T_s\}, \tag{51}$$

where

$$T_d = \begin{cases} T_i - T_o, \text{ if } i \neq 5 \ (T_i \text{ is the temperature at the } i^{th} \text{ point}) \\ T_{avg} - T_o, \text{ if } i = 5 \ (T_{avg} \text{ is the average temperature over the four grid points}) \end{cases} \cdot \tag{52}$$

4.87.12 The TØRDRG Element

4.87.12.1 Input Data for the TØRDRG Element

1. The ECPT/EST entries for the axisymmetric toroidal ring (TØRDRG) element are:

Symbol	Description
SIL_1, SIL_2	Scalar index numbers for the two grid points
α_1, α_2	Angles of curvature at the two grid points (degrees)
γ	Material property orientation angle (currently not used)
H_m, H_f	Membrane and flexure thickness
N_1, X_1, Y_1, Z_1 N_2, X_2, Y_2, Z_2	Local coordinate system number and location in basic coordinates of the grid points.
t_μ	Element temperature for material properties

For this element Y_i must equal zero for i = 1 and 2, and we define:

$$\{R\} = \begin{Bmatrix} X_1 \\ X_2 \end{Bmatrix} , \tag{1}$$

$$\{Z\} = \begin{Bmatrix} Z_1 \\ Z_2 \end{Bmatrix} . \tag{2}$$

2. Coordinate system data

The location (X_i, Y_i, Z_i) and local coordinate system number (N_i) of each grid point are used to calculate the 3 by 3 global-to-basic coordinate system transformation matrices, $[T_i]$, i = 1, 2.

3. Material data

The material property input for the TØRDRG element is the same as that for TRIARG element (see section 4.87.10.1) with the following notational changes:

$$E_p = E_r, \quad E_T = E_\theta, \quad \nu_{PT} = \nu_{r\theta}$$

$$\{ALF\} = \begin{Bmatrix} \alpha_r \\ \alpha_\theta \end{Bmatrix}$$

4.87.12.2 General Calculations

1. Compute the following constants used in stiffness matrix generation:

$$C = E_p/E_T \, , \tag{3}$$

$$D_M = E_p \, H_m/(C - \nu_{PT}^2) \, , \tag{4}$$

$$D_B = E_p \, H_f^3/(12 \, (C - \nu_{PT}^2)). \tag{5}$$

2. Determine if the element is a toroidal ring, a conical ring, a cylindrical ring, or a shell cap:

(1) if $\alpha_1 \neq \alpha_2$, then the element is a toroidal ring.

(2) if $\alpha_1 = \alpha_2 = 90^o$, then the element is a cylindrical ring.

(3) if $\alpha_1 = \alpha_2 \neq 90^o$, then the element is a conical ring.

(4) if $\alpha_1 = 0$, then the element is a shell cap.

3. Compute the local coordinate constants for a toroidal element:

$$\phi_\beta = \alpha_2 - \alpha_1 \, , \tag{6}$$

$$R_p = \frac{[(R_2 - R_1)^2 + (Z_2 - Z_1)^2]^{1/2}}{2 \sin (\frac{\phi_\beta}{2})} \, , \tag{7}$$

$$\lambda_1 = \frac{1}{R_p} \, , \tag{8}$$

$$S = (\alpha_2 - \alpha_1)R_p, \tag{9}$$

$$B_\beta = R_1 + R_p\left[\sin\left(\alpha_1 + \frac{\phi_\beta}{2}\right) - \sin(\alpha_1)\right], \tag{10}$$

$$R_T = \frac{B_\beta}{\sin\left(\alpha_1 + \frac{\phi_\beta}{2}\right)}, \tag{11}$$

$$\psi_1 = \cos\left(\alpha_1 + \frac{\phi_\beta}{2}\right), \tag{12}$$

$$\psi_2 = -\frac{\sin\left(\alpha_1 + \frac{\phi_\beta}{2}\right)}{R_p}. \tag{13}$$

4. Compute the local coordinate constants for a conical or cylindrical ring

$$S = [(R_2 - R_1)^2 + (Z_2 - Z_1)^2]^{1/2}, \tag{14}$$

$$B_\beta = R_1 + \frac{S\cos\alpha_1}{2}, \tag{15}$$

$$R_T = \frac{B_\beta}{\sin\alpha_1}, \tag{16}$$

$$R_p = 0, \tag{17}$$

$$\lambda_1 = 0, \tag{18}$$

$$\psi_1 = \cos\alpha_1, \tag{19}$$

$$\psi_2 = 0. \tag{20}$$

5. Generate the transformation matrix from field coordinates to grid point degrees of freedom:

$$[\Gamma_{\beta q}] = \begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{-10}{s^3} & \frac{-6}{s^2} & 0 & \frac{-3}{2s} & 0 & 0 & \frac{10}{s^3} & \frac{-4}{s^2} & 0 & \frac{1}{2s} \\
0 & 0 & \frac{15}{s^4} & \frac{8}{s^3} & 0 & \frac{3}{2s^2} & 0 & 0 & \frac{-15}{s^4} & \frac{7}{s^3} & 0 & \frac{-1}{s^2} \\
0 & 0 & \frac{-6}{s^5} & \frac{-3}{s^4} & 0 & \frac{-1}{2s^3} & 0 & 0 & \frac{6}{s^5} & \frac{-3}{s^4} & 0 & \frac{1}{2s^3} \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{-3}{s^2} & 0 & 0 & 0 & \frac{-2}{s} & 0 & \frac{3}{s^2} & 0 & 0 & 0 & \frac{-1}{s} & 0 \\
\frac{2}{s^3} & 0 & 0 & 0 & \frac{1}{s^2} & 0 & \frac{-2}{s^3} & 0 & 0 & 0 & \frac{1}{s^2} & 0
\end{bmatrix}_{10x12} \quad . \quad (21)$$

6. Generate the transformation matrix from local to system coordinates:

$$[\Gamma_{rs}] = \begin{bmatrix}
\cos\alpha_1 & 0 & -\sin\alpha_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\sin\alpha_1 & 0 & \cos\alpha_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \cos\alpha_2 & 0 & -\sin\alpha_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \sin\alpha_2 & 0 & \cos\alpha_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}_{12\times12} . \qquad (22)$$

7. Rearrange the transformation matrix $[\Gamma_{\beta q}]$ such that the membrane and flexure terms are reversed.

$$[\bar{\Gamma}_{\beta q}] = \begin{bmatrix} \Gamma_{\beta q}^{(2)} \\ \Gamma_{\beta q}^{(1)} \end{bmatrix}_{10 \times 12} , \qquad (23)$$

where $[\Gamma_{\beta q}^{(1)}]$ is the first six rows of $[\Gamma_{\beta q}]$ and $[\Gamma_{\beta q}^{(2)}]$ is the last four rows of $[\Gamma_{\beta q}]$.

4.87.12.3 Integral Calculations

The method used to compute the integrals depends on the geometric shape of the element.

1. Toroidal ring - basic integrals

There are six basic integrals to be computed, of which the first three can best be evaluated by series expansion, but the last three require numerical integration.

$$I_1^j = R_p^{j+1} \int_0^{\phi_\beta} \phi^j d\phi = \frac{R_p^{j+1} \phi_\beta^{j+1}}{j+1} , \tag{24}$$

$$I_2^j = R_p^{J+1} \int_0^{\phi_\beta} \phi^j \sin\phi d\phi = R_p^{j+1} \sum_{i=0}^{\infty} \frac{(-1)^i \phi_\beta^{j+1+2i+1}}{(j+1+2i+1)\ (2i+1)!} , \tag{25}$$

$$I_3^j = R_p^{j+1} \int_0^{\phi_\beta} \phi^j \cos\phi d\phi = R_p^{j+1} \sum_{i=0}^{\infty} \frac{(-1)^i \phi_\beta^{j+1+2i}}{(j+1+2i)\ (2i)!} , \tag{26}$$

$$I_4^j = R_p^{j+1} \int_0^{\phi_\beta} \frac{\phi^j \sin^2\phi}{B} d\phi , \tag{27}$$

$$I_5^j = R_p^{j+1} \int_0^{\phi_\beta} \frac{\phi^j 2\sin\phi\cos\phi}{B} d\phi , \tag{28}$$

$$I_6^j = R_p^{j+1} \int_0^{\phi_\beta} \frac{\phi^j \cos^2\phi}{B} d\phi , \tag{29}$$

where

$$B = R_1 - R_p \sin\alpha_1 + R_p \sin\alpha_1 \cos\phi + R_p \cos\alpha_1 \sin\phi, \tag{30}$$

and $j = 0, 1, \ldots, 10$.

The summations in I_2^j and I_3^j are carried out until the truncation error is insignificant.

2. Toroidal ring - required integrals

The actual integrals required can now be defined in terms of basic integrals.

$$\delta_1^j = (R_1 - R_p\sin\alpha_1)I_1^j + R_p\cos\alpha_1 I_2^j + R_p\sin\alpha_1 I_3^j , \tag{31}$$

$$\delta_2^j = \cos\alpha_1 I_2^j + \sin\alpha_1 I_3^j , \tag{32}$$

$$\delta_3^j = \cos^2\alpha_1 I_4^j + \sin\alpha_1\cos\alpha_1 I_5^j + \sin^2\alpha_1 I_6^j , \tag{33}$$

$$\delta_4^j = \cos\alpha_1 I_3^j - \sin\alpha_1 I_2^j , \tag{34}$$

$$\delta_5^j = \sin\alpha_1\cos\alpha_1 (I_6^j - I_4^j) + \frac{1}{2}(1 - 2\sin^2\alpha_1) I_5^j , \tag{35}$$

$$\delta_6^j = \cos^2\alpha_1 I_6^j - \sin\alpha_1\cos\alpha_1 I_5^j + \sin^2\alpha_1 I_4^j , \tag{36}$$

$$j = 0,1,\ldots,10$$

3. Conic ring - basic integrals

$$I_1^j = \int_0^S \xi^j d\xi = \frac{S^{j+1}}{j+1} , \quad j = 0,1,\ldots,10 \tag{37}$$

$$I_2^j = \int_0^S \left(\frac{\xi^j}{R_1+\xi\cos\alpha_1}\right) d\xi , \quad j = 0,1,\ldots,10 \tag{38}$$

$$I_2^0 = \frac{1}{\cos\alpha_1} \ln\left(\frac{R_1+S\cos\alpha_1}{R_1}\right), \tag{39}$$

$$I_2^1 = \frac{1}{\cos\alpha_1}\left[S - \frac{R_1}{\cos\alpha_1} \ln\left(\frac{R_1+S\cos\alpha_1}{R_1}\right)\right], \tag{40}$$

$$I_2^j = \frac{S^{j+1}}{R_1} \sum_{i=0}^{\infty} \frac{(-1)^i \left|\frac{S\cos\alpha_1}{R_1}\right|^i}{(j+1+i)} , \quad j = 2,3,\ldots,10 . \tag{41}$$

4. Conic ring - required integrals

$$\left.\begin{aligned}
\delta_1^{\,j} &= R_1 I_1^{\,j} + \psi_1 I_1^{\,j+1}, && (42)\\
\delta_2^{\,j} &= \sin\alpha_1 I_1^{\,j}, && (43)\\
\delta_3^{\,j} &= \sin^2\alpha_1 I_2^{\,j}, && (44)\\
\delta_4^{\,j} &= \psi_1 I_1^{\,j}, && (45)\\
\delta_5^{\,j} &= \sin\alpha_1\, \psi_1 I_2^{\,j}, && (46)\\
\delta_6^{\,j} &= \psi_1^{\,2} I_2^{\,j}, && (47)
\end{aligned}\right\} \quad j = 0,1,\ldots,10$$

5. Cylindrical ring - basic integrals

$$I_1^{\,j} = \int_0^S \xi^j d\xi = \frac{S^{j+1}}{j+1}, \qquad j = 0,1,\ldots,10 \tag{48}$$

$$I_2^{\,j} = \int_0^S \frac{\xi^j}{R_1}\, d\xi = \frac{1}{R_1}\left(\frac{S^{j+1}}{j+1}\right), \quad j = 0,1,\ldots,10 \tag{49}$$

6. Cylindrical ring - required integrals

$$\left.\begin{aligned}
\delta_1^{\,j} &= R_1 I_1^{\,j} + \psi_1 I_1^{\,j+1}, && (50)\\
\delta_2^{\,j} &= \sin\alpha_1 I_1^{\,j}, && (51)\\
\delta_3^{\,j} &= \sin^2\alpha_1 I_2^{\,j}, && (52)\\
\delta_4^{\,j} &= \delta_5^{\,j} = \delta_6^{\,j} = 0, && (53)
\end{aligned}\right\} \quad j = 0,1,\ldots,10$$

4.87.12.4 Elastic Constants Matrix Calculations

Form elastic constants matrix

$$[E] = \frac{1}{\left(1 - \frac{E_T}{E_P}\nu_{PT}^2\right)} \begin{bmatrix} E_P & E_T\nu_{PT} \\ E_T\nu_{PT} & E_T \end{bmatrix}_{2x2} \tag{54}$$

4.87.12.5 Stiffness Matrix Calculations (Subroutine KTØRDR of Module SMA1)

1. Define the constants

$$A = R_p \, , \tag{55}$$

$$V = \nu_{PT} \, , \tag{56}$$

$$C = E_p/E_T \, . \tag{57}$$

2. Form the stiffness matrix terms in field coordinates as shown in Tables 3, 4 and 5.

3. Transform the stiffness matrix from field coordinates to grid point degrees of freedom:

$$[\bar{K}] = [\Gamma_{\beta q}]^T[\tilde{K}][\Gamma_{\beta q}]. \tag{58}$$

4. Transform the stiffness matrix from local to system coordinates:

$$[K] = [\Gamma_{rs}]^T[\bar{K}][\Gamma_{rs}]. \tag{59}$$

5. The global-to-basic coordinate system transformation matrices $[T_i]$ are expanded to 6 by 6 matrices:

$$[T_i^6] = \left[\begin{array}{c|c} T_i & 0 \\ \hline 0 & I \end{array}\right] \, , \; i = 1,2. \tag{60}$$

Table 3. Columns 1, 2 and 3 of the Symmetric 10 by 10 Stiffness Matrix for the TØRDRG Element.

	Column 1	Column 2	Column 3
Row 1	$D_M(\frac{C}{A^2}\delta_1^0 + \frac{2V}{A}\delta_2^0 + \delta_3^0)$		
Row 2	$D_M(\frac{C}{A^2}\delta_1^1 + \frac{2V}{A}\delta_2^1 + \delta_3^1)$	$D_B\delta_6^0 + D_M(\frac{C}{A^2}\delta_1^2 + \frac{2V}{A}\delta_2^2 + \delta_3^2)$	
Row 3	$D_M(\frac{C}{A^2}\delta_1^2 + \frac{2V}{A}\delta_2^2 + \delta_3^2)$	$D_B(2V\delta_4^0 + 2\delta_6^1) + D_M(\frac{C}{A^2}\delta_1^3 + \frac{2V}{A}\delta_2^3 + \delta_3^3)$	$D_B4(C\delta_1^0 + 2V\delta_4^1 + \delta_6^2 + D_M(\frac{C}{A^2}\delta_1^4 + \frac{2V}{A}\delta_2^4 + \delta_3^4)$
Row 4	$D_M(\frac{C}{A^2}\delta_1^3 + \frac{2V}{A}\delta_2^3 + \delta_3^3)$	$D_B(6V\delta_4^1 + 3\delta_6^2) + D_M(\frac{C}{A^2}\delta_1^4 + 2\frac{V}{A}\delta_2^4 + \delta_3^4)$	$D_B6(2C\delta_1^1 + 3V\delta_4^2 + \delta_6^3) + D_M(\frac{C}{A^2}\delta_1^5 + \frac{2V}{A}\delta_2^5 + \delta_3^5)$
Row 5	$D_M(\frac{C}{A^2}\delta_1^4 + \frac{2V}{A}\delta_2^4 + \delta_3^4)$	$D_B(12V\delta_4^2 + 4\delta_6^3) + D_M(\frac{C}{A^2}\delta_1^5 + \frac{2V}{A}\delta_2^5 + \delta_3^5)$	$D_B2(12C\delta_1^2 + 16V\delta_4^3 + 4\delta_6^4) + D_M(\frac{C}{A^2}\delta_1^6 + \frac{2V}{A}\delta_2^6 + \delta_3^6)$
Row 6	$D_M(\frac{C}{A^2}\delta_1^5 + \frac{2V}{A}\delta_2^5 + \delta_3^5)$	$D_B(20V\delta_4^3 + 5\delta_6^4) + D_M(\frac{C}{A^2}\delta_1^6 + 2\frac{V}{A}\delta_2^6 + \delta_3^6)$	$D_B10(4C\delta_1^3 + 5V\delta_4^4 + \delta_6^5) + D_M(\frac{C}{A^2}\delta_1^7 + \frac{2V}{A}\delta_2^7 + \delta_3^7)$
Row 7	$D_M(\frac{V}{A}\delta_4^0 + \delta_5^0)$	$D_M(\frac{V}{A}\delta_4^1 + \delta_5^1)$	$D_M(\frac{V}{A}\delta_4^2 + \delta_5^2)$
Row 8	$D_M(\frac{C}{A}\delta_1^0 + \frac{V}{A}\delta_4^1 + V\delta_2^0 + \delta_5^1)$	$D_M(\frac{C}{A}\delta_1^1 + \frac{V}{A}\delta_4^2 + V\delta_2^1 + \delta_5^2)$	$D_M(\frac{C}{A}\delta_1^2 + \frac{V}{A}\delta_4^3 + V\delta_2^2 + \delta_5^3)$
Row 9	$D_M(\frac{2C}{A}\delta_1^1 + \frac{V}{A}\delta_4^2 + 2V\delta_2^1 + \delta_5^2)$	$D_M(2\frac{C}{A}\delta_1^2 + \frac{V}{A}\delta_4^3 + 2V\delta_2^2 + \delta_5^3)$	$D_M(2\frac{C}{A}\delta_1^3 + \frac{V}{A}\delta_4^4 + 2V\delta_2^3 + \delta_5^4$
Row 10	$D_M(\frac{3C}{A}\delta_1^2 + \frac{V}{A}\delta_4^3 + 3V\delta_2^2 + \delta_5^3)$	$D_M(\frac{3C}{A}\delta_1^3 + \frac{V}{A}\delta_4^4 + 3V\delta_1^3 + \delta_5^4)$	$D_M(\frac{3C}{A}\delta_1^4 + \frac{V}{A}\delta_4^5 + 3V\delta_2^4 + \delta_5^5)$

Table 4. Columns 4, 5, and 6 of the Symmetric 10 by 10 Stiffness Matrix for the TØRDRG Element.

	Column 4	Column 5	Column 6
Row 4	$D_B 9(4C\delta_1^2 + 4V\delta_4^3 + \delta_6^4)$ + $D_M(\frac{C}{A^2}\delta_1^6 + \frac{2V}{A}\delta_2^6 + \delta_3^6)$		
Row 5	$D_B 12(6C\delta_1^3 + 5V\delta_4^4 + \delta_6^5)$ + $D_M(\frac{C}{A^2}\delta_1^7 + \frac{2V}{A}\delta_2^7 + \delta_3^7)$	$D_B 16(9C\delta_1^4 + 6V\delta_4^5 + \delta_6^6)$ + $D_M(\frac{C}{A^2}\delta_1^8 + \frac{2V}{A}\delta_2^8 + \delta_3^8)$	
Row 6	$D_B 15(8C\delta_1^4 + 6V\delta_4^5 + \delta_6^6)$ + $D_M(\frac{C}{A^2}\delta_1^8 + \frac{2V}{A}\delta_2^8 + \delta_3^8)$	$D_B 20(12C\delta_1^5 + 7V\delta_4^6 + \delta_6^7)$ + $D_M(\frac{C}{A^2}\delta_1^9 + \frac{2V}{A}\delta_2^9 + \delta_3^9)$	$D_B 25(16C\delta_1^6 + 8V\delta_4^7 + \delta_6^8)$ + $D_M(\frac{C}{A^2}\delta_1^{10} + \frac{2V}{A}\delta_2^{10} + \delta_3^{10})$
Row 7	$D_M(\frac{V}{A}\delta_4^3 + \delta_5^3)$	$D_M(\frac{V}{A}\delta_4^4 + \delta_5^4)$	$D_M(\frac{V}{A}\delta_4^5 + \delta_5^5)$
Row 8	$D_M(\frac{C}{A}\delta_1^3 + \frac{V}{A}\delta_4^4 + V\delta_2^3 + \delta_5^4)$	$D_M(\frac{C}{A}\delta_1^4 + \frac{V}{A}\delta_4^5 + V\delta_2^4 + \delta_5^5)$	$D_M(\frac{C}{A}\delta_1^5 + \frac{V}{A}\delta_4^6 + V\delta_2^5 + \delta_5^6)$
Row 9	$D_M(\frac{2C}{A}\delta_1^4 + \frac{V}{A}\delta_4^5 + 2V\delta_2^4 + \delta_5^5)$	$D_M(\frac{2C}{A}\delta_1^5 + \frac{V}{A}\delta_4^6 + 2V\delta_2^5 + \delta_5^6)$	$D_M(\frac{2C}{A}\delta_1^6 + \frac{V}{A}\delta_4^7 + 2V\delta_2^6 + \delta_5^7)$
Row 10	$D_M(\frac{3C}{A}\delta_1^5 + \frac{V}{A}\delta_4^5 + 3V\delta_2^5 + \delta_5^5)$	$D_M(\frac{3C}{A}\delta_1^6 + \frac{V}{A}\delta_4^7 + 3V\delta_2^6 + \delta_5^7)$	$D_M(\frac{3C}{A}\delta_1^7 + \frac{V}{A}\delta_4^8 + 3V\delta_2^7 + \delta_5^8)$

Table 5. Columns 7, 8, 9, and 10 of the Symmetric 10 by 10 Stiffness Matrix for the TØRDRG Element.

	Col. 7	Col. 8	Col. 9	Col. 10
Row 7	$D_M\delta_6^0$			
Row 8	$D_M(V\delta_4^0 + \delta_6^1)$	$D_M(C\delta_1^0 + 2V\delta_4^1 + \delta_6^2)$		
Row 9	$D_M(2V\delta_4^1 + \delta_6^2)$	$D_M(2C\delta_1^1 + 3V\delta_4^2 + \delta_6^3)$	$D_M(4C\delta_1^2 + 4V\delta_4^3 + \delta_6^4)$	
Row 10	$D_M(3V\delta_4^2 + \delta_6^3)$	$D_M(3C\delta_1^2\ 4V\delta_4^3 + \delta_6^4)$	$D_M(6C\delta_1^3 + 5V\delta_4^4 + \delta_6^5)$	$D_M(9C\delta_1^4 + 6V\delta_4^5 + \delta_6^6)$

6. The stiffness matrix, [K], is partitioned into 6 by 6 matrices:

$$[K] \Longrightarrow \left[\begin{array}{c|c} K^6_{11} & K^6_{12} \\ \hline K^6_{21} & K^6_{22} \end{array}\right] . \tag{61}$$

7. These 6x6 partitions are transformed to local coordinates:

$$[K^6_{pj}]_{\ell} = [T^6_p]^T [K^6_{pj}][T^6_j], \tag{62}$$

where j = 1, 2, and p is the pivot point, p = 1 or 2.

4.87.12.6 Mass Matrix Calculations (Subroutine MTØRDR of Module SMA2)

1. Form the mass matrix in field coordinates:

$$[\tilde{M}] = 2\pi\rho H_m \begin{bmatrix} \delta_1^0 & & & & & & & & & \\ \delta_1^1 & \delta_1^2 & & & & & & & & \\ \delta_1^2 & \delta_1^3 & \delta_1^4 & & & \text{Symmetric} & & & & \\ \delta_1^3 & \delta_1^4 & \delta_1^5 & \delta_1^6 & & & & & & \\ 0 & 0 & 0 & 0 & \delta_1^0 & & & & & \\ 0 & 0 & 0 & 0 & \delta_1^1 & \delta_1^2 & & & & \\ 0 & 0 & 0 & 0 & \delta_1^2 & \delta_1^3 & \delta_1^4 & & & \\ 0 & 0 & 0 & 0 & \delta_1^3 & \delta_1^4 & \delta_1^5 & \delta_1^6 & & \\ 0 & 0 & 0 & 0 & \delta_1^4 & \delta_1^5 & \delta_1^6 & \delta_1^7 & \delta_1^8 & \\ 0 & 0 & 0 & 0 & \delta_1^5 & \delta_1^6 & \delta_1^7 & \delta_1^8 & \delta_1^9 & \delta_1^{10} \end{bmatrix}_{10x10} \quad (63)$$

2. Transform the mass matrix to grid point degrees of freedom:

$$[\bar{M}] = [\bar{\Gamma}_{\beta q}]^T [\tilde{M}][\bar{\Gamma}_{\beta q}] \quad . \quad (64)$$

3. Transform the mass matrix from local to system coordinates:

$$[M] = [\Gamma_{rs}]^T [\bar{M}][\Gamma_{rs}] \quad . \quad (65)$$

4. The mass matrix, [M], is partitioned into 6x6 matrices, and these 6x6 partitions are transformed to local coordinates (see Equations 61 and 62).

4.87.12.7 Thermal Load Calculations (Subroutine TTØRDR of Module SSG1)

1. Compute the temperature gradient constants:

$$\Delta T_1^{(m)} = T_1^{(m)} - T_o \ , \tag{66}$$

$$\Delta T_2^{(m)} = T_2^{(m)} - T_1^{(m)} \ , \tag{67}$$

$$\Delta T_1^{(f)} = 0 \ , \tag{68}$$

$$\Delta T_2^{(f)} = 0 \ , \tag{69}$$

where $T_i^{(m)}$ are the membrane temperatures at point i.

2. Compute the thermal strain vectors:

$$\{\varepsilon_T^{(0)}\} = \Delta T_i^{(m)}\{ALF\} \ , \tag{70}$$

$$\{\varepsilon_T^{(1)}\} = \Delta T_2^{(m)}\{ALF\} \ , \tag{71}$$

$$\{H_T^{(0)}\} = \Delta T_1^{(f)}\{ALF\} \ , \tag{72}$$

$$\{H_T^{(1)}\} = \Delta T_2^{(f)}\{ALF\} \ , \tag{73}$$

where {ALF} is a vector of length two; the first component is α_r and the second α_θ.

3. Form the matrices of integrals

$$[\tilde{F}_{ME}^{(0)}] = 2\pi H_m \begin{bmatrix} 0 & \delta_1^0 & 2\delta_1^1 & 3\delta_1^2 & \lambda_1\delta_1^0 & \lambda_1\delta_1^1 & \lambda_1\delta_1^2 & \lambda_1\delta_1^3 & \lambda_1\delta_1^4 & \lambda_1\delta_1^5 \\ \delta_4^0 & \delta_4^1 & \delta_4^2 & \delta_4^3 & \delta_2^0 & \delta_2^1 & \delta_2^2 & \delta_2^3 & \delta_2^4 & \delta_2^5 \end{bmatrix}_{2x10} \tag{74}$$

$$[\tilde{F}_{ME}(1)] = \frac{2\pi H_m}{S} \begin{bmatrix} 0 & \delta_1^1 & 2\delta_1^2 & 3\delta_1^3 & \lambda_1\delta_1^1 & \lambda_1\delta_1^2 & \lambda_1\delta_1^3 & \lambda_1\delta_1^4 & \lambda_1\delta_1^5 & \lambda_1\delta_1^6 \\ \delta_4^1 & \delta_4^2 & \delta_4^3 & \delta_4^4 & \delta_2^1 & \delta_2^2 & \delta_2^3 & \delta_2^4 & \delta_2^5 & \delta_2^6 \end{bmatrix}_{2x10} \quad (75)$$

$$[\tilde{F}_{FE}(0)] = \frac{2\pi H_f^3}{12} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -2\delta_1^0 & -6\delta_1^1 & -12\delta_1^2 & -20\delta_1^3 \\ 0 & 0 & 0 & 0 & 0 & -\delta_4^0 & -2\delta_4^1 & -3\delta_4^2 & -4\delta_4^3 & -5\delta_4^4 \end{bmatrix}_{2x10} \quad (76)$$

$$[\tilde{F}_{FE}(1)] = \frac{2\pi H_f^3}{12s} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -2\delta_1^1 & -6\delta_1^2 & -12\delta_1^3 & -20\delta_1^4 \\ 0 & 0 & 0 & 0 & 0 & -\delta_4^1 & -2\delta_4^2 & -3\delta_4^3 & -4\delta_4^4 & -5\delta_4^5 \end{bmatrix}_{2x10} \quad (77)$$

4. Compute the thermal load vector in field coordinates

$$\{\tilde{F}_T\} = [\tilde{F}_{ME}(0)]^T[E]\{\varepsilon_T(0)\} + [\tilde{F}_{ME}(1)]^T[E]\{\varepsilon_T(1)\} + [\tilde{F}_{FE}(0)]^T[E]\{H_T(0)\} + [\tilde{F}_{FE}(1)]^T[E]\{H_T(0)\} . \quad (78)$$

5. Transform the thermal load vector to grid point degrees of freedom:

$$\{\bar{F}_T\} = [\Gamma_{\beta q}]^T\{\tilde{F}_T\} . \quad (79)$$

6. Transform the thermal load vector from local to system coordinates:

$$\{F_T\} = [\Gamma_{rs}]^T\{\bar{F}_T\} . \quad (80)$$

7. Transform the thermal load vector to basic coordinates:

$$\{F_T\}_b = [T_{12}]^T\{F_T\}, \quad (81)$$

where

$$[T_{12}] = \begin{bmatrix} T_1 & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & T_2 & 0 \\ 0 & 0 & 0 & I \end{bmatrix} , \tag{82}$$

and $[T_i]$ and $[I]$ are 3 by 3 matrices.

8. These vectors are added to the overall load vector, $\{P_g\}$.

4.87.12.8 Element Force and Stress Calculations (Subroutines STØRD1 and STØRD2 of Module SDR1)

Element stress and force data are calculated in two phases. The first phase (subroutine STØRD1) calculates the element stiffness and stress matrices. The second phase (subroutine STØRD2) calculates the element forces and stresses from the various subcase displacement vectors. Stresses are evaluated at both ends and at the mid-span of the element.

Phase 1 calculations are as follows:

1. Form the element stiffness matrix, [K], as in section 4.87.12.5.

2. Set up the coordinates of the three stress points:

$$\{X\} = \begin{Bmatrix} 0 \\ \frac{S}{2} \\ S \end{Bmatrix} . \tag{83}$$

3. Compute the constants λ_2, λ_3, λ_4 as a function of the stress points coordinates.

   a. If the element is a toroidal ring, then:

$$\lambda_2^{(i)} = \frac{\cos\left(\alpha_1 + \frac{X(i)}{R_p}\right)}{R_1 - R_p\left[\sin\alpha_1 - \sin\left(\alpha_1 + \frac{X(i)}{R_p}\right)\right]} , \tag{84}$$

$$\lambda_3^{(i)} = \frac{\sin\left(\alpha_1 + \frac{X(i)}{R_p}\right)}{R_1 - R_p\left[\sin\alpha_1 - \sin\left(\alpha_1 + \frac{X(i)}{R_p}\right)\right]} \quad i = 1,2,3, \tag{85}$$

$$\lambda_4^{(i)} = -\lambda_3^{(i)}/R_p \ . \tag{86}$$

b. If the element is a cylindrical or conical ring, then:

$$\lambda_2^{(i)} = \frac{\cos\alpha_1}{R_1 + X(i)\cos\alpha_1} \ , \tag{87}$$

$$\lambda_3^{(i)} = \frac{\cos\alpha_1}{R_1 + X(i)\cos\alpha_1} \quad i = 1,2,3, \tag{88}$$

$$\lambda_4^{(i)} = 0 \ . \tag{89}$$

c. If the element is a shell cap, then:

$$\lambda_2^{(i)} = \frac{\cos\left(\alpha_1 + \frac{X(i)}{R_p}\right)}{R_1 - R_p\left[\sin\alpha_1 - \sin\left(\alpha_1 + \frac{X(i)}{R_p}\right)\right]} \quad i = 1,2,3 \ , \tag{90}$$

$$\lambda_3^{(i)} = 1/R_p \ , \tag{91}$$

$$\lambda_4^{(i)} = -1/(R_p^2) \ . \tag{92}$$

4. Compute the stress matrix in field coordinates for the three stress points as shown in Tables 6 and 7. Note that the factors H and $H^3/12$ have been omitted for $[\tilde{S}_1^{(i)}]$ and $[\tilde{S}_2^{(i)}]$ respectively.

If the element is a shell cap, then modify $[\tilde{S}^{(i)}]$ by:

$$\tilde{S}_{12} = E_{12} + E_{11} \ , \tag{93}$$

$$\tilde{S}_{22} = E_{22} + E_{12} \ , \tag{94}$$

$$\tilde{S}_{37} = -2\ (E_{12} + E_{11}) \ , \tag{95}$$

$$\tilde{S}_{47} = 2\ (E_{22} + E_{12}) \ , \tag{96}$$

$$\tilde{S}_{58} = 3(E_{22} - 4\ E_{11}) \ . \tag{97}$$

5. Form the master stress matrix in field coordinates:

$$[\tilde{S}] = \begin{bmatrix} \tilde{S}^{(1)} \\ \hline \tilde{S}^{(2)} \\ \hline \tilde{S}^{(3)} \end{bmatrix} \quad 15 \times 10 \ . \tag{98}$$

Table 6. Terms in Columns 1 Through 6 of the 2 by 10 $[\tilde{S}_1^{(i)}]$ Matrix and the 3 by 10 $[\tilde{S}_2^{(i)}]$ Matrix.

The $[\tilde{S}_1^{(i)}]$ Matrix

Col. 1	Col. 2	Col. 3	Col. 4	Col. 5	Col. 6
$\lambda_2^{(i)}E_{12}$	E_{11} $+\lambda_2^{(i)}E_{12}X(i)$	$2E_{11}X(i)$ $+\lambda_2^{(i)}E_{12}X^2(i)$	$3E_{11}X^2(i)$ $+\lambda_2^{(i)}E_{12}X^3(i)$	$\lambda_1^{(i)}E_{11}$ $+\lambda_3^{(i)}E_{12}$	$(\lambda_1^{(i)}E_{11}$ $+\lambda_3^{(i)}E_{12})X(i)$
$\lambda_2^{(i)}E_{22}$	E_{12} $+\lambda_2^{(i)}E_{22}X(i)$	$2E_{12}X^2(i)$ $+\lambda_2^{(i)}E_{22}X^2(i)$	$3E_{12}X^2(i)$ $+\lambda_2^{(i)}E_{22}X^3(i)$	$\lambda_1^{(i)}E_{12}$ $+\lambda_3^{(i)}E_{22}$	$(\lambda_1^{(i)}E_{12}$ $+\lambda_3^{(i)}E_{22})X(i)$

The $[\tilde{S}_2^{(i)}]$ Matrix

0	0	0	0	0	$-\lambda_2^{(i)}E_{12}$
0	0	0	0	0	$\lambda_2^{(i)}E_{22}$
0	0	0	0	0	$g^{(i)}$

where $g^{(i)} = (\lambda_2^{(i)})^2 E_{22} - \lambda_4^{(i)}E_{12}$ and

$$[\tilde{S}^{(i)}] = \begin{bmatrix} \tilde{S}_1^{(i)} \\ \hline \tilde{S}_2^{(i)} \end{bmatrix}_{5\times10}$$

Table 7. Terms in Columns 7 Through 10 of the 2 by 10 $[\tilde{S}_1^{(i)}]$ Matrix and the 3 by 10 $[\tilde{S}_2^{(i)}]$ Matrix.

The $[\tilde{S}_1^{(i)}]$ Matrix

	Col. 1	Col. 2	Col. 3	Col. 4
Row 1	$(\lambda_1^{(i)}E_{11}$	$(\lambda_1^{(i)}E_{11}$	$(\lambda_1^{(i)}E_{11}$	$(\lambda_1^{(i)}E_{11}$
	$+ \lambda_3^{(i)}E_{12}X^2(i)$	$+ \lambda_3^{(i)}E_{12})X^3(i)$	$+ \lambda_3^{(i)}E_{12})X^4(i)$	$+ \lambda_3^{(i)}E_{12})X^5(i)$
Row 2	$(\lambda_1^{(i)}E_{12}$	$(\lambda_1^{(i)}E_{12}$	$(\lambda_1^{(i)}E_{12}$	$(\lambda_1^{(i)}E_{12}$
	$+ \lambda_3^{(i)}E_{22})X^2(i)$	$+ \lambda_3^{(i)}E_{22})X^3(i)$	$+ \lambda_3^{(i)}E_{22})X^4(i)$	$+ \lambda_3^{(i)}E_{22})X^5(i)$

The $[\tilde{S}_2^{(i)}]$ Matrix

Row 1	$-2\lambda_2^{(i)}E_{12})X(i)$	$- 3\lambda_2^{(i)}E_{12}X^2(i)$	$- 4\lambda_2^{(i)}E_{12}X^3(i)$	$- 5\lambda_2^{(i)}E_{12}X^4(i)$
	$- 2E_{11}$	$- 6E_{11}X(i)$	$- 12E_{11}X^2(i)$	$- 20E_{11}\ X^3(i)$
Row 2	$2\lambda_2^{(i)}E_{22}X(i)$	$3\lambda_2^{(i)}E_{12}X^2(i)$	$4\lambda_2^{(i)}E_{22}X^3(i)$	$5\lambda_2^{(i)}E_{22}X^4(i)$
	$+ 2E_{12}$	$+ 6E_{12}X(i)$	$+ 12E_{12}X^2(i)$	$+ 20E_{12}X^3(i)$
Row 3	$2g^{(i)}X(i)$	$3g^{(i)}X^2(i)$	$4g^{(i)}X^3(i)$	$5g^{(i)}X^4(i)$
	$- 2\lambda_2^{(i)}E_{11}$	$- 6\lambda_2^{(i)}E_{11}X(i)$	$- 12\lambda_2^{(i)}E_{11}X^2(i)$	$- 20\lambda_2^{(i)}E_{11}X^3(i)$
		$- 6E_{11}$	$- 24E_{11}X(i)$	$- 60E_{11}X^2(i)$

where $g^{(i)} = (\lambda_2^{(i)})^2 E_{22} - \lambda_4^{(i)}E_{12}$ and

$$[\tilde{S}^{(i)}] \quad \left[\begin{array}{c} \tilde{S}_1^{(i)} \\ \hline \tilde{S}_2^{(i)} \end{array}\right]$$

6. Transform the stress matrix to grid point degrees of freedom

$$[\bar{S}] = [\tilde{S}][\bar{\Gamma}_{\beta q}] . \tag{99}$$

7. Transform the stress matrix from local to system coordinates

$$[S] = [\bar{S}]\,[\Gamma_{rs}] . \tag{100}$$

8. Transform the stress matrix to global coordinates

$$[S]_g = [S]\,[T_{12}] . \tag{101}$$

9. Compute the thermal stress vector for the three stress points:

$$\{T_s^{(i)}\} = \begin{Bmatrix} \Delta T_1^{(m)} H_m (E_{11}\alpha_1 + E_{12}\alpha_2) + \Delta T_2^{(m)} H_m \frac{Xi}{S} (E_{11}\alpha_1 + E_{12}\alpha_2) \\ \Delta T_1^{(m)} H_m (E_{21}\alpha_1 + E_{22}\alpha_2) + \Delta T_2^{(m)} H_m \frac{Xi}{S} (E_{21}\alpha_1 + E_{22}\alpha_2) \\ \Delta T_1^{(f)} \frac{H_f^3}{12} (E_{11}\alpha_1 + E_{12}\alpha_2) + \Delta T_2^{(f)} \frac{H_f^3 Xi}{12S} (E_{11}\alpha_1 + E_{12}\alpha_2) \\ -\Delta T_1^{(f)} \frac{H_f^3}{12} (E_{21}\alpha_1 + E_{22}\alpha_2) - \Delta T_2^{(f)} \frac{H_f^3 Xi}{12S} (E_{21}\alpha_1 + E_{22}\alpha_2) \\ \Delta T_1^{(f)} \frac{H_f^3}{12} \lambda_2^{(i)} [(E_{11} - E_{12})\alpha_1 + (E_{12} - E_{22})\alpha_2] \\ \quad + \Delta T_2^{(f)} \frac{H^3}{12S} \Big\{ \lambda_2^{(i)} Xi [(E_{11} - E_{12})\alpha_1 \\ \quad + (E_{12} - E_{22})\alpha_2] + [E_{11}\alpha_1 + E_{12}\alpha_2] \Big\} \end{Bmatrix} \quad (5 \times 1) \tag{102}$$

where $\Delta T_1^{(m)}$, $\Delta T_2^{(m)}$, $\Delta T_1^{(f)}$ and $\Delta T_2^{(f)}$ are as given in Equations 66 through 69.

10. Form the master thermal stress vector

$$\{T_s\} = \begin{Bmatrix} T_s^{(1)} \\ \hline T_s^{(2)} \\ \hline T_s^{(3)} \end{Bmatrix} . \tag{103}$$

Phase 2 calculations are as follows:

1. Extract the displacement vector, $\{\Delta\}$, at the two grid points from the global displacement vector.

2. Calculate the element forces:

$$\{P\} = [K]\{\Delta\} . \tag{104}$$

3. Calculate the element stresses without regard to thermal loading:

$$\{\sigma'\} = [S]_g \{\Delta\} \quad (12 \times 1) . \tag{105}$$

4. If there is no thermal loading, then

$$\{\sigma\} = \{\sigma'\} . \tag{106}$$

5. If there is thermal loading, then

$$\sigma_j = \sigma'_j - (T_1 - T_o)\, T_{S_j} - (T_2 - T_1)\, T_{S_{j+15}} . \tag{107}$$

for j = 1 and 2 and

$$\sigma_j = \sigma'_j . \tag{108}$$

for j = 3, 4, 5.

4.87.13 The VISC Element

The viscous element, VISC, has the same properties as the RØD element (see section 4.87.1) except that instead of generating a contribution to the stiffness matrix, the element generates a contribution to the damping matrix, $[B_{gg}]$.

4.87.13.1 Input Data for the VISC Element

1. The ECPT/EST entries for the VISC are:

Symbol	Description
SIL_a, SIL_b	Scalar indices for grid points a and b
N_a, X_a, Y_a, Z_a N_b, X_b, Y_b, Z_b	Local coordinate system number and basic coordinates of grid points.
C_1	Extensional damping coefficient
C_2	Torsional damping coefficient

Given N_a, X_a, Y_a, Z_a, N_b, X_b, Y_b and Z_b and the CSTM (Coordinate System Transformation Matrices) data block, the 3 by 3 transformation matrices, $[T_a]$ and $[T_b]$, are calculated using utility routine TRANSD (see section 3.4.37).

4.87.13.2 Damping Matrix Calculations (Subroutine BVISC of Module SMA2)

1. Generate {n} as in Equation 2, section 4.87.1.2, and generate the transformation matrices $[T_a]$ and $[T_b]$.

2. Calculate:

$$[b_t] = C_1 \begin{bmatrix} n_1^2 & n_1n_2 & n_1n_3 \\ n_1n_2 & n_2^2 & n_2n_3 \\ n_1n_3 & n_2n_3 & n_3^2 \end{bmatrix} , \qquad (1)$$

$$[b_r] = C_2 \begin{bmatrix} n_1^2 & n_1n_2 & n_1n_3 \\ n_1n_2 & n_2^2 & n_2n_3 \\ n_1n_3 & n_2n_3 & n_3^2 \end{bmatrix} . \quad (2)$$

3. The 6x6 damping matrices are:

$$[B_{aa}] = \begin{bmatrix} T_a^T b_t T_a & 0 \\ 0 & T_a^T b_r T_a \end{bmatrix} , \quad (3)$$

$$[B_{ab}] = - \begin{bmatrix} T_a^T b_t T_b & 0 \\ 0 & T_a^T b_r T_b \end{bmatrix} , \quad (4)$$

$$[B_{bb}] = \begin{bmatrix} T_b^T b_t T_b & 0 \\ 0 & T_b^T b_r T_b \end{bmatrix} , \quad (5)$$

$$[B_{ba}] = - \begin{bmatrix} T_b^T b_t T_a & 0 \\ 0 & T_b^T b_r T_a \end{bmatrix} . \quad (6)$$

4. These matrices are added to $[B_{gg}]$.

4.87.14 Integral Calculations for the TRIARG, TRAPRG Elements

Integrals of the form

$$I_{pq} = \int_{R_j}^{R_i} \int_{Z_{mn}}^{Z_{k\ell}} r^p z^q \, dz dr \quad . \tag{1}$$

must be calculated for the TRIARG and TRAPRG elements (see sections 4.87.10.3 and 4.87.11.3). The integration may be performed for any integer values of p and q. The area of integration is defined by the two lines $r = R_i$ and $r = R_j$, and by the two lines $z = b_{k\ell} r + a_{k\ell}$ and $z = b_{mn} R + a_{mn}$.

This integration is performed by the integration "driver" subroutine, FØRTRAN function DKI in module SMA1, FØRTRAN function DMI in module SMA2, and FØRTRAN function AI in module SDR2.

The following input data are necessary for these routines:

- p - an integer that defines the power of the r variable.
- q - an integer that defines the power of the z variable.
- {R} - a vector of the r coordinates of all points used to describe the area of integration.
- {Z} - a vector of the z coordinates of all points used to describe the area of integration.
- k,ℓ - the subcripts of R, Z defining one of the lines of the limit of integration (i.e., the line between points (r_k, z_k) and (r_ℓ, z_ℓ)).
- m,n - the subscripts of R, Z defining the second line on the limit of integration.
- i,j - the subscripts of R defining the other two lines on the limit of integration.

In the following paragraphs FØRTRAN names of functions auxiliary to DKI are given. The corresponding FØRTRAN function names auxiliary to functions DMI and AI can be found in sections 4.28.8 and 4.46.8 respectively.

The following slopes and y-intercepts are calculated in functions DKK and DKM

$$a_{k\ell} = \frac{R_\ell Z_k - R_k Z_\ell}{R_\ell - R_k} , \tag{2}$$

$$b_{k\ell} = \frac{Z_\ell - Z_k}{R_\ell - R_k} , \tag{3}$$

$$a_{mn} = \frac{R_n Z_m - R_m Z_n}{R_n - R_m} , \tag{4}$$

$$b_{mn} = \frac{Z_n - Z_m}{R_n - R_m} . \tag{5}$$

A test for a vanishing area of integration is made: if $R_i = R_j$, then $I_{pq} = 0$; if $a_{k\ell} = a_{mn}$ and $b_{k\ell} = b_{mn}$, then $I_{pq} = 0$.

The formulas for evaluation of the integrals are dependent upon the values of p and q as given in the following sections.

4.87.14.1 Integral Calculation for $q \geq 0$ and any p. (Function DKINT)

Define the function

$$f_1(x,y) = \frac{1}{q+1} \sum_{t=0}^{q+1} C\, x^t\, y^{q+1-t} D \quad , \tag{6}$$

where

$$C = \begin{cases} \prod_{s=1}^{t} \dfrac{q+1-s+1}{s} & \text{for } t \neq 0 \quad , \\ 1 & \text{for } t = 0 \quad , \end{cases} \tag{7}$$

$$D = \begin{cases} \left[\dfrac{R_j^{(q+1+p+1-t)} - R_i^{(q+1+p+1-t)}}{q+1+p+1-t}\right] & \text{for } (q+1+p+1-t) \neq 0 \\ \ln (R_j/R_i) & \text{for } (q+1+p+1-t) = 0 \end{cases} . \tag{8}$$

C and D are calculated in functions DKEF and DK1 respectively.

The integral is

$$I_{pq} = f_1(a_{mn},b_{mn}) - f_1(a_{k\ell},b_{k\ell}) \quad . \tag{9}$$

4.87.14.2 Integral Calculation for $p \geq 0$ and $q < -1$ (Function DK89)

$$f_2(\alpha,x,y) = \frac{1}{y^{p+1}} \sum_{s=0}^{p} p!(-x)^s D \quad , \tag{10}$$

where

$$D = \begin{cases} \dfrac{(x+yR_\alpha)^{p+1+q+1-s}}{(p-s)!s!(p+1+q+1-s)} & \text{for } (p+1+q+1-s) \neq 0 \\ \dfrac{\ln|x+yR_\alpha|}{(p+1+q+1)!(-q-2)!} & \text{for } (p+1+q+1-s) = 0 \end{cases} , \tag{11}$$

and α = i or j.

The integral is

$$I_{pq} = \frac{1}{2+q}\left[f_2(i,a_{k\ell},b_{k\ell}) - f_2(i,a_{mn}\ b_{mn})\right. \qquad (12)$$

$$\left. - f_2(j,a_{k\ell},b_{k\ell}) + f_2(j,a_{mn},b_{mn})\right]$$

4.87.14.3 Integral Calculation for p< 0 and q< - 1 (Function DK100)

Define the function

$$f_3(\alpha,x,y) = \frac{-1}{x^{(-p-q-2)}} \sum_{s=0}^{-p-q-3} (-p-q-3)\ !\ D\ , \qquad (13)$$

where

$$D = \begin{cases} \dfrac{(x+yR_\alpha)^{(-p-1-s)}\ (-y)^s}{(-p-q-3-s)!s!(-p-1-s)R_\alpha^{(-p-1-s)}} & \text{for } (-p-1-s) \neq 0 \\ \dfrac{(-y)^{-p-1}\ \ln\left(\left|\dfrac{x+yR_\alpha}{R_\alpha}\right|\right)}{(-p-1)!(-q-2)!} & \text{for } (-p-1-s) = 0 \end{cases} , \qquad (14)$$

and α = i or j.

The integral is

$$I_{pq} = \frac{1}{2+q}\left[f_3(i,a_{k\ell},b_{k\ell}) - f_3(i,a_{mn},b_{mn})\right. \qquad (15)$$

$$\left. - f_3(i,a_{k\ell},b_{k\ell}) + f_3(i,a_{mn},b_{mn})\right]$$

4.87.14.4 Integral Calculations for $p > -1$ and $q = -1$ (Function DKJAB)

Define the function

$$f_4(\alpha,x,y) = \frac{R_\alpha^{p+1}\ln(|x+yR_\alpha|)}{p+1} - \frac{y}{p+1} f_2(\alpha,x,y) , \tag{16}$$

where f_2 is given in Equation 10.

The integral is

$$\begin{aligned} I_{pq} = {} & f_4(i,a_{k\ell},b_{k\ell}) - f_4(i,a_{mn},b_{mn}) \\ & - f_4(j,a_{k\ell},b_{k\ell}) - f_4(j,a_{mn},b_{mn}) \end{aligned} . \tag{17}$$

4.87.14.5 Integral Calculations for $p < -1$ and $q = -1$ (Function DK219)

Define the function

$$f_5(\alpha,x,y) = -\frac{\ln(|x+yR_\alpha|)}{(-p-1)R_\alpha^{(-p-1)}} + \frac{y}{(-p-1)} f_3(\alpha,x,y) , \tag{18}$$

where f_3 is given in Equation 13.

The integral is

$$\begin{aligned} I_{pq} = {} & f_5(i,a_{k\ell},b_{k\ell}) - f_5(i,a_{mn},b_{mn}) \\ & - f_5(j,a_{k\ell},b_{k\ell}) - f_5(j,a_{mn},b_{mn}) \end{aligned} . \tag{19}$$

4.87.14.6 Integral Calculations for p = -1 and q = -1 (Function DK211)

Define the function

$$f_6(\alpha,x,y) = \begin{cases} 0\,, & \text{for } yR_\alpha = x \\ \frac{1}{2}\left[\ln(|2\,y\,R_\alpha|)\right]^2\,, & \text{for } (yR_\alpha)^2 = x^2 \text{ and } yR_\alpha \neq x \\ \ln|x|\,\ln|R_\alpha| - \sum_{t=1}^{\infty} \frac{1}{t^2}\left[\frac{-yR_\alpha}{x}\right]^t\,, & \text{for } (yR_\alpha)^2 < x^2\,. \\ \frac{1}{2}\left[\ln(|yR_\alpha|)\right]^2 + \sum_{t=1}^{\infty} \frac{1}{t^2}\left[\frac{-x}{yR_\alpha}\right]^t\,, & \text{for } (yR_\alpha)^2 < x^2 \end{cases} \tag{20}$$

The summation term is calculated until its value becomes less than 1.0×10^{-6}.

The integral is

$$I_{pq} = f_6(i,a_{k\ell},b_{k\ell}) - f_6(i,a_{mn},b_{mn}) - f_6(i,a_{k\ell},b_{k\ell}) - f_6(i,a_{mn},b_{mn}) \tag{21}$$

4.88 DETERMINANT METHOD OF EIGENVALUE EXTRACTION

NASTRAN contains two double precision versions of the determinant method. One version is for the solution of real eigenvalue problems; and one version is for the solution of complex eigenvalue problems. The specifications for both versions are discussed in this document. Real arithmetic is used for the real problem, complex for the complex problem.

4.88.1 Fundamentals of the Determinant Method

The basic notion employed in the determinant method of eigenvalue extraction is very simple. If the elements in a matrix [A] are functions of the operator p, then the determinant of [A] can be expressed as:

$$D([A]) = (p-p_1)\,(p-p_2)\,\ldots\,(p-p_n), \tag{1}$$

where p_1, p_2, p_3 ...p_n are the eigenvalues of the matrix. The value of the determinant vanishes for $p = p_i$, $i = 1,2,\ldots,n$.

In the determinant method, the determinant is evaluated for trail values of p, selected according to some iterative procedure, and a criteria is established to determine when D([A]) is sufficiently small or when p is sufficiently close to an eigenvalue. The eigenvector is then found by solution of the equation:

$$[A]\,\{u\} = 0, \tag{2}$$

with one of the elements of {u} set to unity.

4.88.2 Evaluation of Determinant

The most convenient procedure for evaluating the determinant of a matrix employs the triangular decomposition of the matrix: [A] = [L] [U] where [L] is a lower unit triangle (unit values on the diagonal). The determinant of [A] is equal to the product of the diagonal terms of [U]. The matrix [A] may be expressed as

$$[A] = -p[M] + [K], \tag{3}$$

for real eigenvalue problems and as

$$[A] = p^2[M] + p[B] + [K], \tag{4}$$

for complex eigenvalue problems.

4.88.3 Iteration Algorithm

Consider a series of determinants $D_{k-2}^{(i)}$, $D_{k-1}^{(i)}$, $D_k^{(i)}$ evaluated for trial values of the eigenvalue $p = p_{k-2}$, p_{k-1}, p_k. Then a better approximation to the eigenvalue is obtained from the following calculations:

Let

$$h_k = p_k - p_{k-1}, \tag{5}$$

$$\lambda_k = h_k/h_{k-1}, \tag{6}$$

$$\delta_k = 1 + \lambda_k. \tag{7}$$

Then

$$h_{k+1} = \lambda_{k+1} h_k, \tag{8}$$

$$p_{k+1} = p_k + h_{k+1}, \tag{9}$$

where

$$\lambda_{k+1} = \frac{-2 D_k^{(i)} \delta_k}{g_k \pm [g_k^2 - 4D_k^{(i)} \delta_k \lambda_k (D_{k-2}^{(i)} \lambda_k - D_{k-1}^{(i)} \delta_k + D_k^{(i)})]^{1/2}}, \tag{10}$$

$$g_k = D_{k-2}^{(i)} \lambda_k^2 - D_{k-1}^{(i)} \delta_k^2 + D_k^{(i)} (\lambda_k + \delta_k). \tag{11}$$

The (+) or (-) sign in Equation 10 is selected to minimize the absolute value of λ_{k+1}. In the case where p_k, p_{k-1} and p_{k-2} are all aribitrarily selected initial values (starting points), the starting points should be arranged such that $|D_k| \leq |D_{k-1}| \leq |D_{k-2}|$ and the (+) or (-) sign in Equation 10 should be selected to minimize the distance from p_{k+1} to all 3 starting points.

In a real eigenvalue analysis it is possible to calculate a complex λ_{k+1}. In order to

preclude the use of complex arithmetic in a real eigenvalue analysis problem, only the real part of the λ_{k+1} should be used to estimate a p_{k+1}.

4.88.4 Scaling

In calculating the determinant of [A], the determinant is scaled by powers of 10 since the accumulated product will rapidly overflow or underflow the floating point size of a digital computer. All operations using the determinant are calculated in scaled arithmetic.

4.88.5 Sweeping of Previously Extracted Eigenvalues

Once an eigenvalue has been found to satisfactory accuracy, a return to that eigenvalue by the iteration algorithm can be prevented by dividing the determinant by the factor $(p-p_i')$ where p_i' is the accepted approximation to p in all subsequent calculations.

Thus:

$$D^{(1)}([A]) = \frac{D([A])}{p-p_1'} \tag{12}$$

should be used in place of D([A]) after the first eigenvalue has been found. In general, the reduced determinant used for finding the $i+1^{st}$ eigenvalue is:

$$D^{(i)}([A]) = \frac{D^{(i-1)}([A])}{(p-p_i')} = \frac{D([A])}{(p-p_1')(p-p_2')\cdots(p-p_i')} \quad . \tag{13}$$

This sweeping procedure is quite satisfactory provided that all p_i' have been calculated to an accuracy that is limited only by round-off error.

For problems in which zero is an eigenvalue, the number of such eigenvalues is specified by the user. In using the determinant method, zero eigenvalues should be eliminated from the determinant by a preliminary operation,

$$D^{(o)}([A]) = \frac{D([A])}{p^m} \quad , \tag{14}$$

where m is the multiplicity of the zero eigenvalue.

For problems with conjugate complex eigenvalues (complex eigenvalue analysis with real matrices) the conjugates of extracted eigenvalues should also be swept from the determinant. Thus

$$D^{(i)}([A]) = \frac{D^{(i-1)}([A])}{(p-p_i')(p-\bar{p}_i')} \quad , \tag{15}$$

where $\bar{p}_i'$ is the conjugate of p_i'. It should be noted that the sweeping equations are indeterminant for $p = p_i'$. This situation will occur when a root coincides with a starting point or a new estimate with a root already extracted. When the first situation occurs, the starting point should be moved away from the root. When the second situation occurs, D_{k+1} should be set equal to D_k.

4.88.6 Convergence Criteria

Convergence criteria are based on successive values of the increment h_k in the estimated eigenvalue. No tests on the magnitude of the determinant or any of the diagonal terms of the triangular decomposition are necessary or desirable. Wilkinson[1] shows that for h_k sufficiently small, the magnitude of h_k is approximately squared for each successive iteration when using Muller's method. This is an extremely rapid rate of convergence. In a very few iterations the "zone of indeterminacy" is reached within which h_k remains small but exhibits random behavior due to round-off error. Wilkinson states that if it is desired to calculate the root to the greatest possible precision, the convergence criterion for accepting p_k as a root should be:

$$|h_{k+1}| > |h_k|. \tag{16}$$

The determinant method accepts this advice, tempered by practical considerations. The first of these is that Equation 16 may be falsely satisfied during the first few iterations while the root tracking algorithm is picking up the "scent". Thus it must, in addition, be required that $|h_k|$, $|h_{k-1}|$ and $|h_{k-2}|$ be reasonably small. The second practical consideration is that we may waste a few iterations within the zone of indeterminacy while waiting for Equation 16 to be satisfied. This is avoided by accepting p_k if $|h_k|$ is sufficiently small. Finally, if the number of iterations becomes excessively large without satisfying a convergence criterion, the Determinant Method assumes the existence of one iteration loop, gives up and proceeds to

a new set of starting points.

Figure 1 is a flow diagram of a set of tests which meet the requirements discussed above for real eigenvalue problems. The tests are based on calculated values of $\overline{H1}$, $\overline{H2}$, and $\overline{H3}$ which are defined as:

$$\overline{H}_1 = |h_{k-1}| / \sqrt{|p_k|}, \tag{17}$$

$$\overline{H}_2 = |h_k| / \sqrt{|p_k|}, \tag{18}$$

$$\overline{H}_3 = |h_{k+1}| / \sqrt{|p_k|}, \tag{19}$$

where $p_k = k^{th}$ estimate of an eigenvalue and $h_k = p_k - p_{k-1}$.

A similar set of tests is described in Figure 2 for complex eigenvalue problems. In this case $\overline{H1}$, $\overline{H2}$ and $\overline{H3}$ are defined as:

$$\overline{H1} = |h_{k-1}|, \tag{20}$$

$$\overline{H2} = |h_k|, \tag{21}$$

$$\overline{H3} = |h_{k+1}|. \tag{22}$$

The magnitude of the convergence criterion ε should be selected as a compromise between running time and accuracy. Currently $\varepsilon = 10^{-11}$. If failure occurs because the number of iterations exceeds the iteration limit, NIT, for two successive sets of starting points, ε is increased by a factor of 10. If successive pairs of failures still occur, ε is again increased until the number of permissible changes in ε is exceeded. The user is informed of the reduced precision of the calculations.

Since eigenvalues are swept out after they are found, all sets of starting points will eventually lead to failure by the preliminary range checks or through successive iteration failure. When this occurs it is presumed that all eigenvalues within the range of interest have been found and the calculations are halted. If for some reason this does not occur the calculation must still be halted. The one remaining avenue for the computer to continue calculations indefinitely is if it continues to find roots. To block this avenue, the calcula-

tions are stopped if the number of roots found exceeds the maximum number of N_{EVM}. As a safeguard the order in which the roots are found is indicated to the user.

4.88.7 Extraction of Eigenvectors

Once an approximate eigenvalue p_j has been accepted, the eigenvector is determined by back substitution into the previously computed triangular decomposition of $[A(p_j)]$. Now since

$$[A(p_j)]\{u\} = [L(p_j)][U(p_j)]\{u\} = 0, \tag{23}$$

we work only with $[U(p_j)]$. Because partial pivoting (row interchanges) have been used, the last diagonal term in $[U(p_j)]$ will normally be the only term with a very small value. The normal appearance of $[U(p_j)]$ is as follows, for n = 7:

$$\begin{bmatrix} X & X & X & X & 0 & 0 & 0 \\ & X & X & X & X & 0 & 0 \\ & & X & X & X & X & 0 \\ & & & X & X & X & X \\ & 0 & & & X & X & X \\ & & & & & X & X \\ & & & & & & \delta \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix} = \{0\} . \tag{24}$$

The terms in the upper right corner are zero due to bandwidth. δ is a very small number. The eigenvector may be extracted by assigning an arbitrary value (such as 1.0) to u_7 and solving successively for u_6, u_5 etc., from the preceeding rows. Note that this is equivalent to placing a load vector {F} on the right-hand side that is null except for the last term which is set equal to δ.

Situations may occur in which U_{nn} is not the smallest diagonal term. Let U_{ii} be the smallest diagonal term with $i < n$. The most common reason for this occurrence is that the degrees of freedom $u_{i+1}, u_{i+2}, \ldots, u_n$ are, for some reason, uncoupled to the preceeding degrees of freedom. In this case all of the elements in the i^{th} row of $[U(p_j)]$ will be very small as shown for i = 4, n = 7:

$$\begin{bmatrix} X & X & X & X & 0 & 0 & 0 \\ & X & X & X & X & 0 & 0 \\ & & X & X & X & X & 0 \\ & & & \delta_{44} & \delta_{45} & \delta_{46} & \delta_{47} \\ & 0 & & & X & X & X \\ & & & & & X & X \\ & & & & & & X \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix} = \{0\}. \tag{25}$$

In the event of multiple or pathologically close eigenvalues two or more rows of $[U(p_j)]$ will consist of very small values, exhibited below for the very exceptional case where the n^{th} row is not very small:

$$\begin{bmatrix} X & X & X & X & 0 & 0 & 0 \\ & X & X & X & X & 0 & 0 \\ & & X & X & X & X & 0 \\ & & & \delta_{44} & \delta_{45} & \delta_{46} & \delta_{47} \\ & 0 & & & X & X & X \\ & & & & & \delta_{66} & \delta_{67} \\ & & & & & & X \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix} = \{0\}. \tag{26}$$

In order to accommodate the exceptional cases described above with the more general case of $\delta = U_{nn}$, a full load vector $\{F\}$ is used for the eigenvector calculations. The load vector will also contain elements of the same order of magnitude as the smallest diagonal element of the triangularized matrix $[U(p_j)]$ in order to prevent digital overflow when the eigenvector is calculated. In addition, a distinct load vector is formed for each eigenvalue to ensure that independent eigenvectors are calculated for multiple or pathologically close eigenvalue problems. The following equations are used for $\{F\}$. For real eigenvalues, we have

$$F_i = \frac{\delta(-1)^{ij}}{1 + (1 - \frac{i}{n})\, j} \; ; \tag{27}$$

For complex eigenvalues,

$$Re(F_i) = \frac{|\delta| (-1)^{ij}}{1.0 + (1.0 - \frac{i}{n}) j}, \quad (28)$$

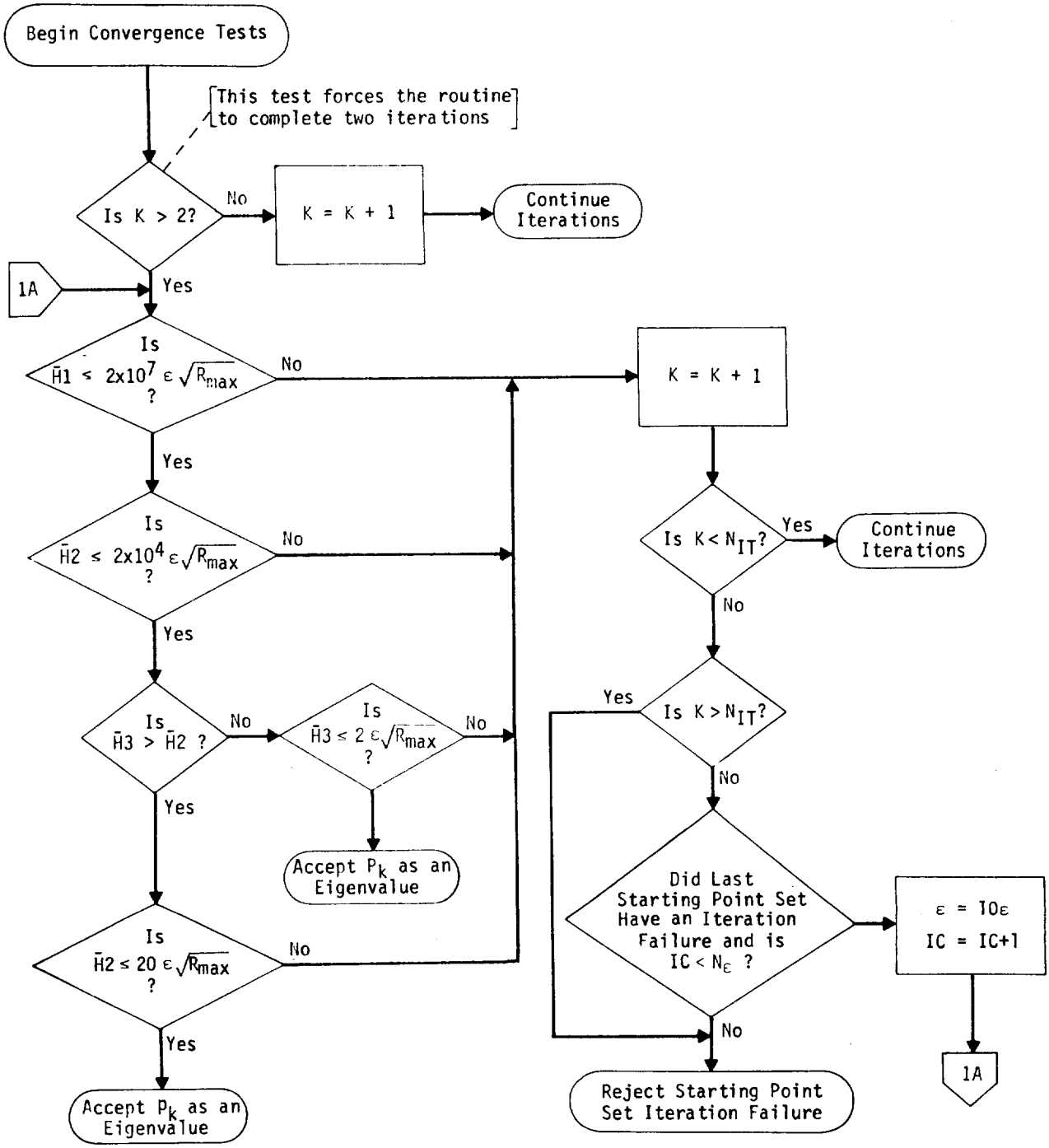
$$Im(F_i) = 0.0, \quad (29)$$

where δ = smallest U_{ii}, j = eigenvalue count, i = i^{th} element of {F} and n = number of rows.

There is a possibility that the smallest diagonal element of [U] may be exactly zero for some eigenvalue. This will be the case when the accepted eigenvalue (p_j) is exactly equal to an eigenvalue of the problem. When this occurs, $\delta = 1.0 \times 10^{-8}$. The calculated eigenvectors are normalized to a unit maximum real element value.

REFERENCE

1. Wilkinson, J.H., "The Algebraic Eigenvalue Problem", Clarendon Press, Oxford, 1965.



ε- Convergence Criterion

K - Iteration Counter

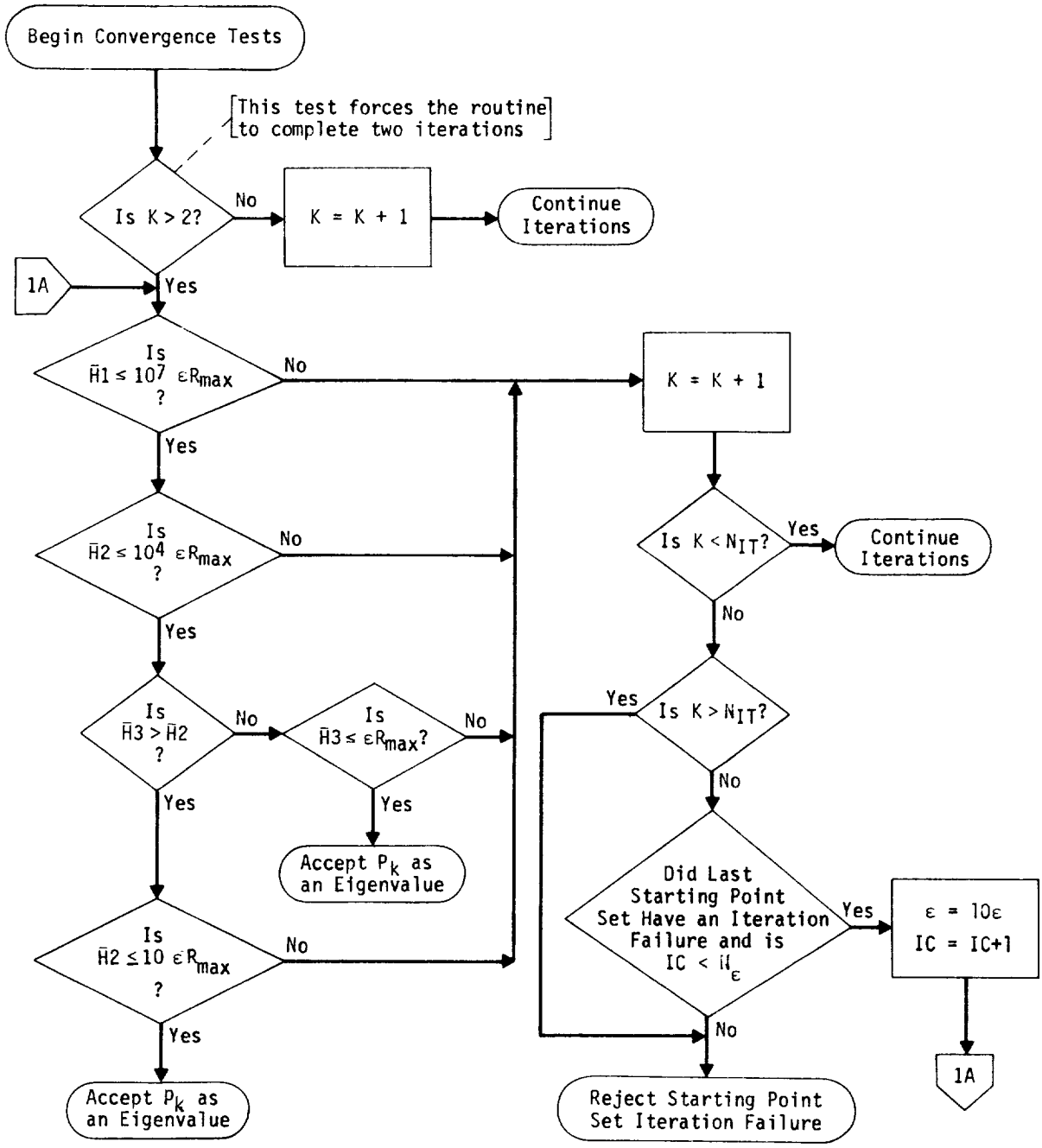
IC - Criterion Change Counter

$\bar{H}1 = |h_{k-1}| / \sqrt{|P_k|}$

$\bar{H}2 = |h_k| / \sqrt{|P_k|}$

$\bar{H}3 = |h_{k+1}| \sqrt{|P_k|}$

Figure 1. Real eigenvalue problems convergence tests



ε- Convergence Criterion

K - Iteration Counter

IC - Criterion Change Counter

$\bar{H}1 = |h_{k-1}|$

$\bar{H}2 = |h_k|$

$\bar{H}3 = |h_{k+1}|$

Figure 2. Complex eigenvalue problems convergence tests

5.1 INTRODUCTION

NASTRAN operates on: 1) the IBM 7094/7040(44) Direct Couple System under the IBSYS operating system; 2) the IBM System/360 under Operating System (ØS); 3) the Univac 1108 under the Exec 8 operating system; and 4) the CDC 6600 under the SCØPE 3 operating system. This section discusses the interfaces between NASTRAN and these operating systems with respect to: 1) input/output; 2) link switching; 3) overlay considerations and implementation of the open core concept; 4) the setup of the operating system control cards preceding (and, in the case of the Univac 1108, following) the NASTRAN data decks; 5) generation of an executable NASTRAN system; and 6) machine dependent routines.

The vocabulary used in each subsection is the one used by systems programmers familiar with the particular operating system being discussed. It is to these system programmers that each subsection is addressed.

5.2 NASTRAN ON THE IBM 7094/7040(44) DCS (IBSYS)

5.2.1 Introduction

The purpose of this section is to describe the features of NASTRAN which are unique to the IBM 7094/7040(44) Direct Couple System (DCS) implementation. The DCS has had a unique place in the design and development of NASTRAN. It is the only second generation machine on which NASTRAN runs. In general its operating system was less able to accomodate a program of the size and complexity of NASTRAN than third generation operating systems, and hence "set the rule" concerning maximum size of programs and links, complexity of overlay, etc. At the same time, it was realized that the DCS was an obsolete system and would eventually be replaced by third generation machines. Therefore, the design of NASTRAN, especially in executing efficiency, was made optimum for the third generation machines often at the expense of performance on the DCS.

Conversely, the DCS was the only system operating with enough reliability to allow effective development when the project began. Hence all routines were initially checked out on the DCS. The DCS remained the development system for the major portion of NASTRAN development. It is the system on which NASTRAN is maintained. The source library is propagated from the DCS to all other machines. In some sense then, familiarity and practical experience on the DCS during the development of NASTRAN has compensated for its second generation nature.

5.2.2 Input/Output

The machine dependent input/output routines on the DCS can be partitioned into 3 classes. Class 1 obtains the available units, classifies them as to disk or tape, and stores pointers to these units in the FIAT Executive table. Class 2 communicates with the 7044(40) to perform the physical I/Ø, manages buffers, handles timing and device errors, etc. Class 3 is actually a modified subset of Class 2 and pertains specifically to the generation of plot tapes.

5.2.2.1 Class 1 - GNFIAT

GNFIAT establishes the equivalence between FIAT entries and the unit control blocks in the system nucleus. Each unit control block (UCB) is checked for availability. If it is available, it is checked against the entries in the permanent FIST. These entries are, in order:

SYSUT1	DATA POOL FILE (PØØL TAPE)
SYSUT5	OLD PROBLEM TAPE
SYSUT6	NEW PROBLEM TAPE
SYSUT7	BCD PLOT TAPE
SYSUT8	BINARY PLOT TAPE
SYSLB2	SUBSYS TAPE
SYSLB3	OLD USER's MASTER FILE (OR SOURCE LIB) TAPE
SYSLB4	NEW USER's MASTER FILE (OR OBJECT LIB) TAPE
SYSUT9	POTENTIAL INPUT TO USER MODULE "INPUT"

If the available unit is not one of the above or is not a physical tape ("set-up"), a pointer to this UCB is placed in the FIAT. If it is a permanent FIST entry and is set up, a pointer to this UCB is placed in the permanent FIAT. Note that certain system units are also exempt from assignment. These are SYSLB1, SYSCRD, SYSPRT, SYSPCH, SYSØUT, SYSIN1 and SYSPP1. If any unit is a physical tape, this fact is noted in the FIAT entry for that unit. The user can cause specific files to be assigned to "free" tapes by use of the FILE DMAP declaration.

5.2.2.2 Class 2 - GINØ

The following decks constitute the assembly language GINØ set on the DCS: ØPENN, CLØSED, READD, WRITEN, FØRWRD, BKREC, SELECT, SKIPF, MISC and SYSUNT. GINØ was written in assembly language to allow several improvements on the normal I/Ø routines available on the DCS. These improvements include: 1) smaller core resident code, 2) variable buffer sizes, and 3) faster execution times. GINØ on the DCS uses IØEX to communicate with the 7094/44. GINØ writes unblocked records with a current maximum record length of 100. This number can be changed by the use of the NASTRAN card as the first card in the Executive Control Deck. Figures 2 through 11 comprise a set of flowcharts for the assembly language GINØ routines for the DCS.

5.2.2.3 Class 3 - SGINØ

In order to write long unformatted records, SGINØ changes the buffer size while it is in control. It sets a flag for BCD or binary mode records. It also causes GINØ to not attach its normal control words. These modifications are necessary to meet plotter hardware requirements. Otherwise, SGINØ calls the normal GINØ entry points of ØPEN, WRITE and CLØSE.

5.2.3 Link Switching

During the execution of a NASTRAN run it often is necessary to call a module which exists in another link of the program. To accomplish this on the DCS, subroutine SEARCH is called.

CALL SEARCH (JØBSEG,FILNAM,REWFLG)

JØBSEG - BCD name of the next link desired - BCD - input.

FILNAM - BCD name of the system unit on which the program links reside - BCD - input.

REWFLG - Flag to indicate whether the program tape should be rewound or not.

1 = Rewind
0 = No rewind

} Integer - input

For example, suppose NASTRAN was currently in link 3 and the XSEM3 subroutine determined that the next module to be executed resided in link 4. The appropriate call would be: CALL SEARCH (6HNSO4 ,6HSYSLB2,0). Note that the links are called NSi, where i = 01,02,..., is the link number. SEARCH is scatter-loaded into high core and reads the program links until a match is found for JØBSEG. The root segment is loaded into core, and SEARCH transfers control to the entry point of the main program in this link. Before SEARCH is called, core resident tables are saved on a empty unit, and the file number of this unit is passed through blank common. This task is accomplished via ENDSYS. After the new link has been loaded, the main program immediately calls BGNSYS to restore these tables.

On the DCS, link switching is an expensive operation since there is no non-transmitting read. Experience has shown that the time to switch from a link to one adjacent to it is 30 seconds with a maximum switch time to any link of 60 seconds. For small problems this builds in a substantial overhead time. However, this time is constant, and its importance decreases as the problem size increases. Unfortunately certain classes of iterative problems (Piecewise Linear Analysis, Static Analysis with Differential Stiffness, flutter problems, etc.) may suffer serious running time additions due to this overhead time.

5.2.4 Overlay Considerations and Implementation of Open Core

The execution time of a NASTRAN run is extremely sensitive to the amounts of core storage available to certain routines. Small differences (1000 or so) in the number of words available can often make differences in running time of factors of 10 or 20.

With these considerations in mind, the overlay for the DCS has been constantly changed to obtain as much data storage space as possible. At the same time, many routines have spill logic built in, but often serious inefficiencies result when this code is called into use. Two major design decisions were forced on NASTRAN to accomodate the small core size of the DCS. One was an assembly language GINØ for the DCS. This allowed considerable savings in code and buffer space over the standard I/Ø routines used in the DCS. For further details on the I/Ø see section 5.2.2. The second was removing the format conversion routines (FCNV,FIØH) from the root segment and allowing only selected modules to reference the system output file.

At the same time, overlaying is not a cheap operation on the DCS. Hence many routines which might only in rare cases require large amounts of core storage were grouped together to minimize the number of overlay segments during execution. The DCS overlay structure was duplicated as nearly as possible on the other machines in the interest of consistency during checkout and simplicity of system maintenance. A pictorial representation of the DCS overlay structure is given in the figures in section 5.2.9.

The only unusual feature of NASTRAN overlay structures on the DCS is caused by the desire of several module drivers to reference several matrix routines in common. This can be best illustrated by the following figure.

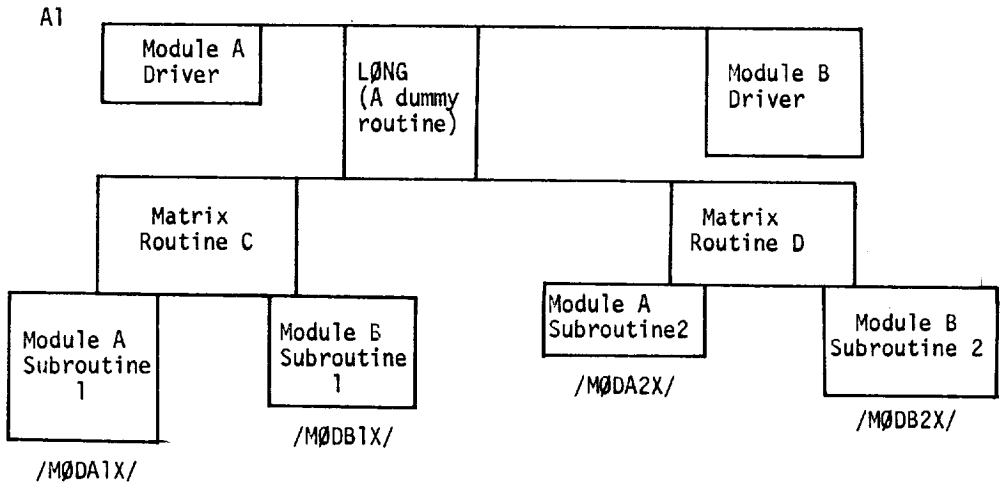


Figure 1. A NASTRAN overlay on the IBM 7094/7040(44) DCS.

Module A and B both wish to call Matrix Routines C and D. To accomplish this, the following calls are made. The module A driver is called and loaded. Module A driver calls Module A Subroutine 1. This call causes the loader to load Module A Subroutine 1, Matrix Routine C and LØNG. LØNG is a DCS only deck, which is never called and which contains a DIMENSIØN statement for a dummy array whose length is longer than the length of all module drivers origined at A1. Hence, the core allocated for the Module A driver is not disturbed. Module A Subroutine 1 returns to Module A driver which calls Module A Subroutine 2 and so on. Thus the drivers for modules A and B have access to Matrix Routines C and D without any loss of core storage to Matrix Routine D when Matrix Routine C is in execution.

The nature of open core on the DCS can also be illustrated by this picture. All core below /MØDA1X/ is available to Module A Subroutine 1 to do with as it pleases. In order to find out how much is available, Module A Subroutine 1 can call function CØRSZ which on the DCS computes the number of words between /MØDA1X/ and blank CØMMØN. Thus all unused (by code) cells of core are dynamically made available to each routine.

Note also that a special version of .LØVRY, a DCS only deck, must be used in conjunction with SEARCH and LNKSTK (see section 5.2.6) in order that the overlays for each link can all be stacked on one logical unit. Further details can be found in the .LØVRY listing.

5.2.5 Execution Deck Setup

The purpose of this section is to describe the system control cards required to execute a NASTRAN run on the DCS. The following list of cards is a general guide to format and type. It is unlikely that any run would use every card listed below. Read the appropriate description section, and choose those cards relevant to your particular run.

Table 1. Execution Deck Setup for NASTRAN on the IBM 7094/7040(44) DCS.

```
$JOB  CSC,J208000,14903-0,S0314,A  IC1,20,20000     NASTRAN RUN                          01
$SETUP LB2      SUBSYS,DISK,COPY,FC=27          THIS PUTS SUBSYS ON DISK             02
$SETUP UT1      POOL            THIS WILL MAKE THE POOL A PHYSICAL TAPE              03
$SETUP UT5      OPTP,NORING     THIS WILL MAKE THE OPTP A PHYSICAL TAPE              04
$SETUP UT6      NPTP            THIS WILL MAKE THE NPTP A PHYSICAL TAPE              05
$SETUP UT7      PLT1            THIS IS THE BCD PLOT TAPE --EAI AND B-L              06
$SETUP UT8      PLT2            THIS IS THE BINARY PLOT TAPE -- SC4020               07
$SETUP UT9      USER,NORING     THIS IS A USER INPUT TAPE                            08
$SETUP LB3      UMF,NORING      THIS IS NEEDED FOR USING A USERS MASTER FILE         09
$SETUP LB4      NUMF            THIS IS USED WHILE CREATING A USERS MASTER FILE      10
$SETUP UT3      TAPE            THIS FILE WILL BE USED TO HONOR TAPE REQUESTS        11
$ASSIGN         SYSLB2                                                               12
$ASSIGN         SYSLB3                                                               13
$ASSIGN         SYSLB4                                                               14
$ASSIGN         SYSUT1                                                               15
$ASSIGN         SYSUT2                                                               16
$ASSIGN         SYSUT3                                                               17
$ASSIGN         SYSUT4                                                               18
$ASSIGN         SYSUT5                                                               19
$ASSIGN         SYSUT6                                                               20
$ASSIGN         SYSUT7                                                               21
$ASSIGN         SYSUT8                                                               22
$ASSIGN         SYSUT9                                                               23
$ASSIGN         SYSUT0                                                               24
$ATEND          0,77777,3                                                            25
$IBJOB CSC      GO,ALTIO,NOFLOW,NOMAP,NOSOURCE                                       26
$IBLDR CALLER       DUMMY MAIN PROGRAM CALLER OBJECT DECK GOES HERE                  27
$IBLDR .SRCH.   SEARCH OBJECT DECK GOES HERE TO BRING IN LINK 1                      28
$DATA                                                                                29
$$$$$$     YOUR NASTRAN DATA DECKS FOLLOW THE $DATA CARD                             30
```

Note that each card has a number in columns 79-80. Each card or group of cards will be described under that number below.

01 - Standard installation accounting card. Certain standard information is contained on this card which will be important to the successful completion of a run.

Time Limit = 20. This number is an absolute and will override any number on the TIME card in the Executive Control Deck.

Number of Lines of output = 20000. Again this number will override any MAXLINES setting in the Case Control Deck.

02 - This causes the program tape to be placed on the disk ready for execution. The program tape can be left as a physical tape but at the expense of execution speed.

03 - For large problems the Data Pool File (PØØL) will contain many millions of words. Especially on systems where the disk is "cleaned up" only at the completion of a job, large problems may need the PØØL set up to save disk space.

04 - When doing a restart, this tape should be the one generated from previous run.

05 - When checkpointing, this card is required. You may not checkpoint onto the disk.

06-07 - One of these cards is required for plots. Both types of plots may be made on one run, in which case both cards are required. If the plotter is "on-line", certain modifications of these cards may be desirable for faster turn-around, but the form shown here is acceptable. The density of these two tapes is not a user option but is often set by the hardware of the plotter.

08 - This tape cannot be used without user programming. Its intent is to allow the user to easily input data from another source into NASTRAN. The user then must provide the module (INPUT) to process this data into NASTRAN format.

09 - The User Master File is for bulk data storage. It functions as one alternate source of input along with the ØPTP and the card reader.

10 - If it is desired to create or edit a User Master File, this tape should be set up.

11 - SYSUT2-4 are available for use in conjunction with the TAPE option in the FILE DMAP instruction.

Note: If any cards between 03 and 11 are not being used, they should simply be removed from the deck. NASTRAN will automatically make these units available for later use.

12-24 - Required system ASSIGN cards. Actually, these cards are required only for physical units, but it does not hurt to leave them in permanently.

25 - This card forces a system dump under certain conditions, such as maximum time being exceeded, etc. It may be removed or retained as desired.

26 - This is the standard IBJØB card. Note the NØMAP and NØSØURCE options. Output will start with NASTRAN output, not loader maps.

27-28 - These cards are replaced by the appropriate binary driver decks.

29-30 - The $DATA card is followed by the 3 NASTRAN data decks. The contents of these decks are described in section 2 of the User's Manual.

5.2.6 Generation of the NASTRAN Executable System ("Subsysing").

In order to generate the executable NASTRAN system, the loader must be called in for each link. Unfortunately there is no simple way in IBSYS to join separate outputs from IBLDR. However, a special technique was developed by David E. Bluett, Reference 1, called "subsysing", which allows one to accomplish this task. The routine which does the subsysing is called LNKSTK. Hence, a NASTRAN subsys run consists of 2n (n being the number of NASTRAN links) separate job steps with LNKSTK putting each link of NASTRAN on a tape. The resultant program is linked together through SEARCH at execution time. The following list gives a set of key cards which accomplish the subsysing and gives a brief description of each card.

Table 2. NASTRAN Subsys Deck Setup on the IBM 7094/7040(44) DCS.

```
$JOB  CSC,J208000,14903-0,S0314,A  IC1,90,20000    NASTRAN SUBSYS              01
$SETUP LB2      SUBSYS                  NEW SUBSYS TAPE                     02
$SETUP LB4      OBJL,DISK,COPY,FC=1     OBJECT LIBRARY                      03
$ASSIGN         SYSLB2                                                      04
$ASSIGN         SYSLB3                                                      05
$ASSIGN         SYSLB4                                                      06
$ASSIGN         SYSUT1                                                      07
$ASSIGN         SYSUT2                                              00      08
$ASSIGN         SYSUT3                                                      09
$ASSIGN         SYSUT4                                                      10
$ASSIGN         SYSUT5                                                      11
$ASSIGN         SYSUT6                                                      12
$ASSIGN         SYSUT7                                                      13
$ASSIGN         SYSUT8                                                      14
$ASSIGN         SYSUT9                                                      15
$ASSIGN         SYSUT0                                                      16
$IBJOB KHR      ALTIO,NOFLOW,MAP                                            17
$IEDIT          SYSLB4,SRCH                                                 18
$IBLDR SEMDBD                                                               19
$$      THE REST OF THE ROOT SEGMENT DECKS GO HERE                          20
$ORIGIN         A1,REW,SYSCK1                                               21
$$      THE DECKS FOR OVERLAY AT A1 GO HERE                                 22
$$    THE REST OF THE OVERLAY STRUCTURE IS DESCRIBED HERE                   23
$ENTRY          XSEM1                                                       24
$DATA                                                                       25
@EOF                                                                        26
$IBSYS                                                                      27
$REWIND         SYSLB4                                                      28
$$    LINK 1 HAS JUST BEEN PROCESSED BY IBLDR.                              29
$ID   CSC,J208000,04013-0,46501,A  IC1                                      30
$IBJOB          ALTIO,NOFLOW                                                31
$IEDIT          SYSLB4,SRCH                                                 32
$IBLDR LNKSTK                                                               33
$DATA                                                                       34
NS01  ,SYSCK2,SYSCK1,SYSLB2,SYSLB2,RB                                       35
@EOF                                                                        36
```

Each card in the above listing contains a sequence number in columns 79-80. Each card will be discussed under this heading.

01 - Standard installation dependent accounting card. Note that the subsys time is not small. It normally takes between 45 and 60 minutes on a "dry" system. One link at a time can be generated and merged into an existing subsys tape using the IBEDIT feature of IBSYS.

02 - The subsys tape is set up as a physical tape to ensure recovery if a system failure should abort the run.

03 - The object library contains card-to-tape images of the object decks for all NASTRAN subroutines.

04-16 - System ASSIGN cards.

17 - This is a standard IBJØB with the exception of the NØFLØW option. NASTRAN will not be processed by IBLDR without this option.

18 - This card instructs the Editor to look for the requested decks on SYSLB4.

19-20 - One $IBLDR card is given for each deck required in each link.

21-23 - Normal overlay cards are used to describe the overlay tree. Note that all segments must be routed to SYSCK1 since LNKSTK expects them there.

24 - All main programs (XSEMi's) are subroutines and hence require $ENTRY cards.

25-26 - These cards terminate this job step.

27-29 - SYSLB4 is returned to the beginning to facilitate searching. A message is written on the operator's typewriter to aid in following the job progress.

30-33 - LNKSTK is brought in to take the core resident (root segment) program on SYSCK2 and the overlay segments on SYSCK1 and copy them onto SYSLB2.

34-36 - The data card for LNKSTK describes the position of the overlay tape and the output tape as well as controls the rewind options. Note that every link has a name beginning with NSi, i = 01, 02,....

The above model is repeated for each link that is being subsysed. The final number of files on the program tape is two times the number of links subsysed plus one.

5.2.7 Machine Dependent Routines

The material discussed in this section will be the machine dependent features of certain routines. There are two classes of machine dependent routines. The first class consists of routines whose code is in assembly language and hence varies from machine to machine, but whose function remains exactly the same. Examples of such routines would include the MAPFNS routines: RSHIFT, ANDF, etc. These are documented in subroutine descriptions, section 3. The second class consists of routines which result in different external manifestations on each machine. They are treated below.

5.2.7.1 Clocks

All times measured on the DCS are wall clock times. I/Ø waiting times, overlay times, LINK switching times, etc., are all included. This fact should be born in mind for job time estimation on the DCS. Note also that if any forms of interrupt are used, NASTRAN will not realize that it has not had sole use of the machine. The support functions such as blocking and/or unblocking, plotting, printing, etc., will also reduce the efficiency of the DCS to run NASTRAN. This sort of activity can cause large variances in running times even for the same problem. A request to run "dry" or "moderately loaded" is usually a must for an efficient and predictable NASTRAN run.

5.2.7.2 DUMP/PDUMP

During the normal execution of a NASTRAN run, all calls to DUMP or PDUMP are blocked by the routine KRDUMP. KRDUMP will give a "backtrace" of the deck name, line number, and location for each deck called. This is accomplished by subroutine XRTLIN. This should be enough to allow some error detection. For example, if the following backtrace was given:

```
XSEM2  CALLED SMA1 AT IFN 74
SMA1   CALLED KCØNE AT IFN 421
KCØNE  CALLED MESAGE AT IFN 114,
```

one could tenatively conclude that the Structural Matrix Assembler module (SMA1) encountered some unexpected difficulty in processing the CØNE element.

Often an inspection of the data, along with the error message printed out, immediately resolves the difficulty. In case a full dump of the 7094 core is desired, a DIAG 1 card in

the Executive Control Deck should be used. The resulting dump will be in the "STR" format.

5.2.7.3 CØNMSG

During the execution of a NASTRAN run, all messages to the user which are of an informative nature are printed at the beginning of the job output. These messages include the start and end time for each module, matrix decomposition parameters, warning messages, and any fatal errors detected by NASTRAN. This is accomplished by subroutine CØNMSG automatically. If in addition the user expects to attend the computer run and wishes to follow the progress of NASTRAN (especially for long runs), a DIAG 5 card in the Executive Control Deck will cause the beginning time for each module to be typed on the operator's typewriter. Similarly, a DIAG 6 card will cause the ending time for each module to be typed.

5.2.8 Flow Charts for the DCS Machine Language GINØ Routines

Figures 2 through 11 on the following pages show the flow of the assembly language GINØ routines for the DCS.

Note:

Many "flag" tests shown on the following flowcharts indicate an apparent closed loop. These are "waits" at positions within the logic where processing cannot continue until a particular condition (filled/emptied buffer, file re-positioned, etc.) is satisfied. When the necessary condition is satisfied, IØEX will interrupt any processing (possibly a "wait" loop) and the appropriate flag will be changed. The interrupted processing is then resumed at the point of interrupt. This flag change, if holding a loop, will cause the "wait" to terminate. All loops and their tested flags operate similarly and asynchronously. This interrupt capability of IØEX permits I/Ø and computational overlap.

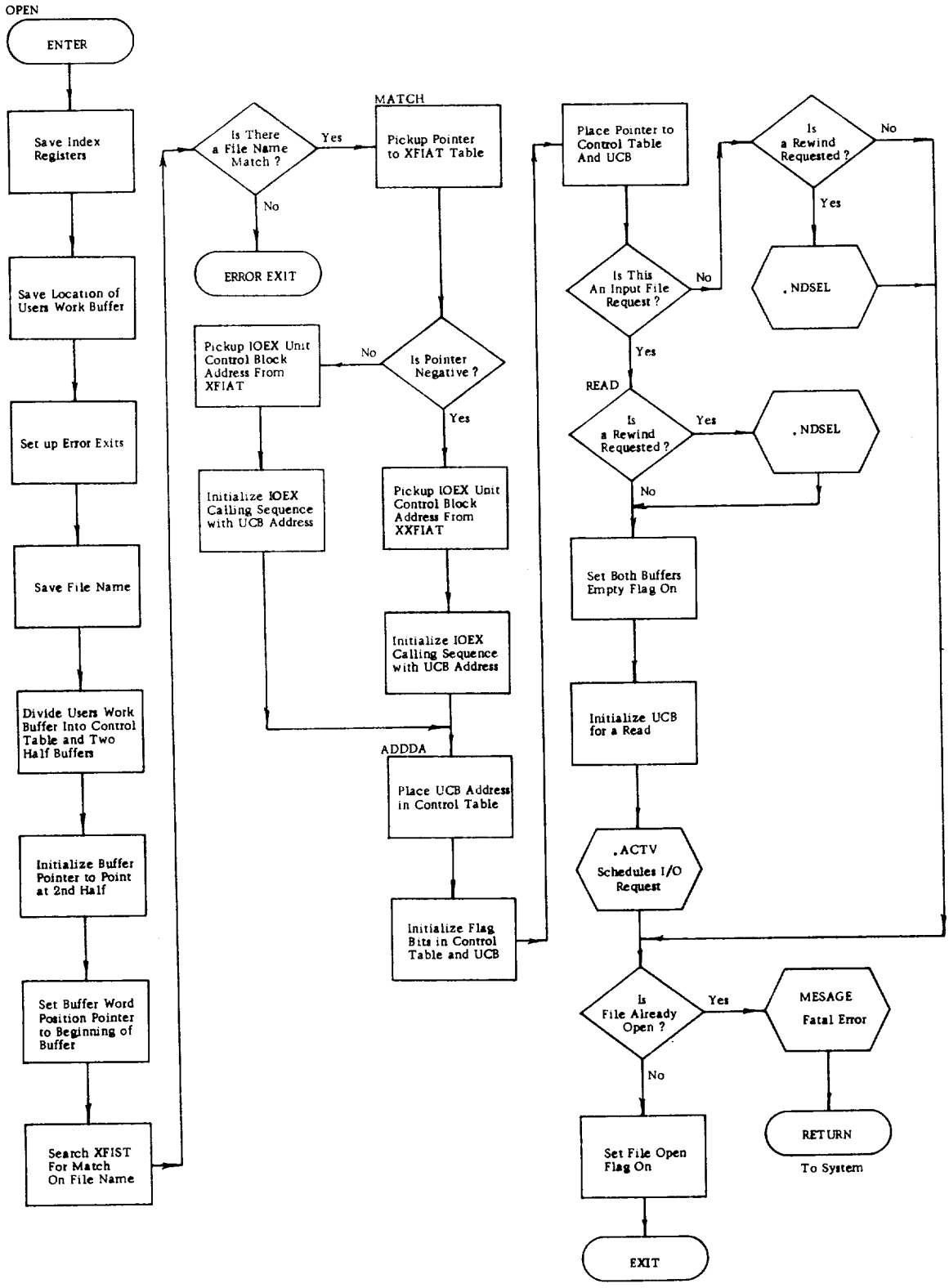


Figure 2. Flowchart for the DCS assembly language GINØ routine ØPEN - deck name ØPENN.

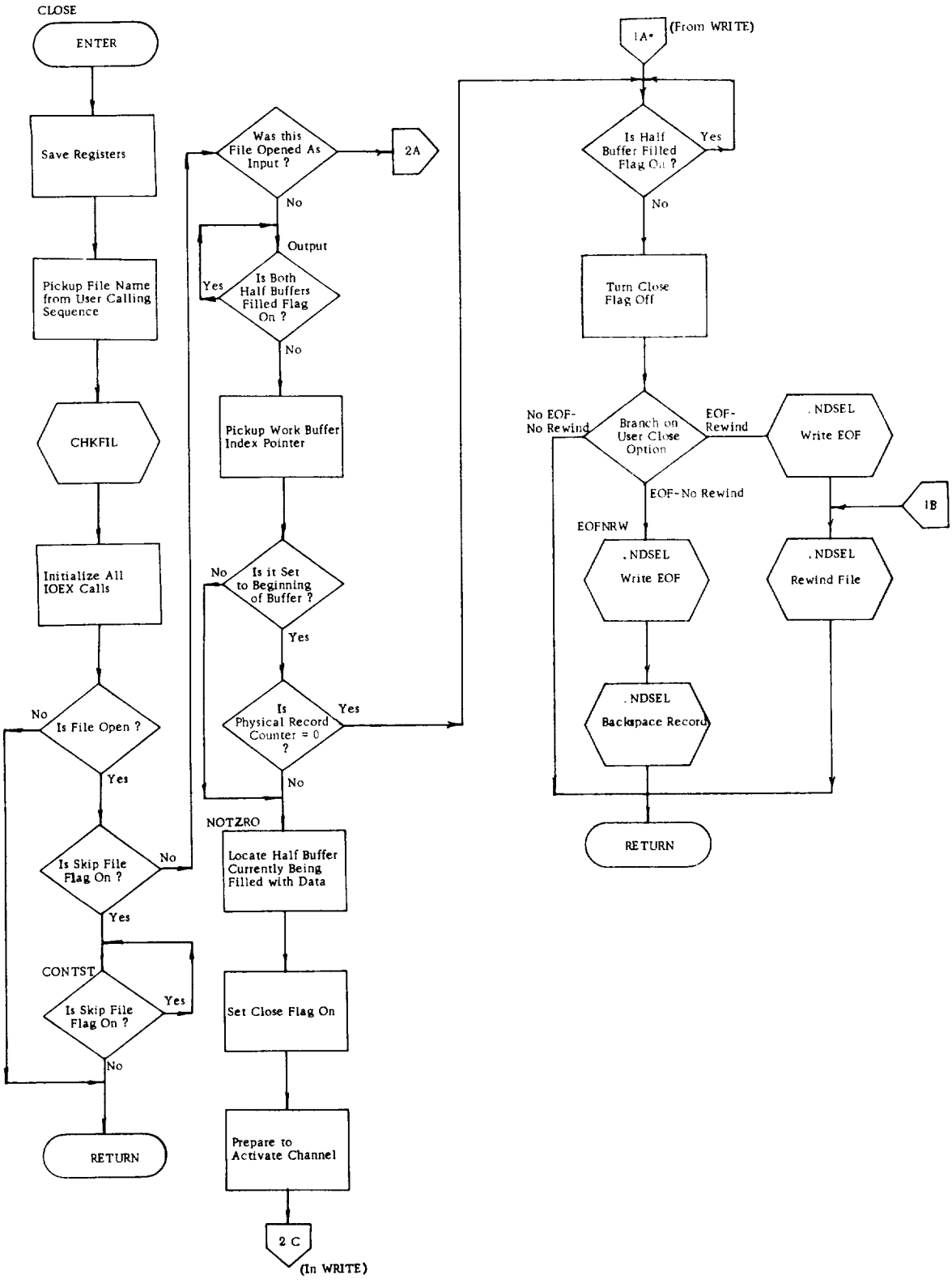


Figure 3(a). Flowchart for the DCS assembly language GINØ routine CLØSE - deck name CLØSED.

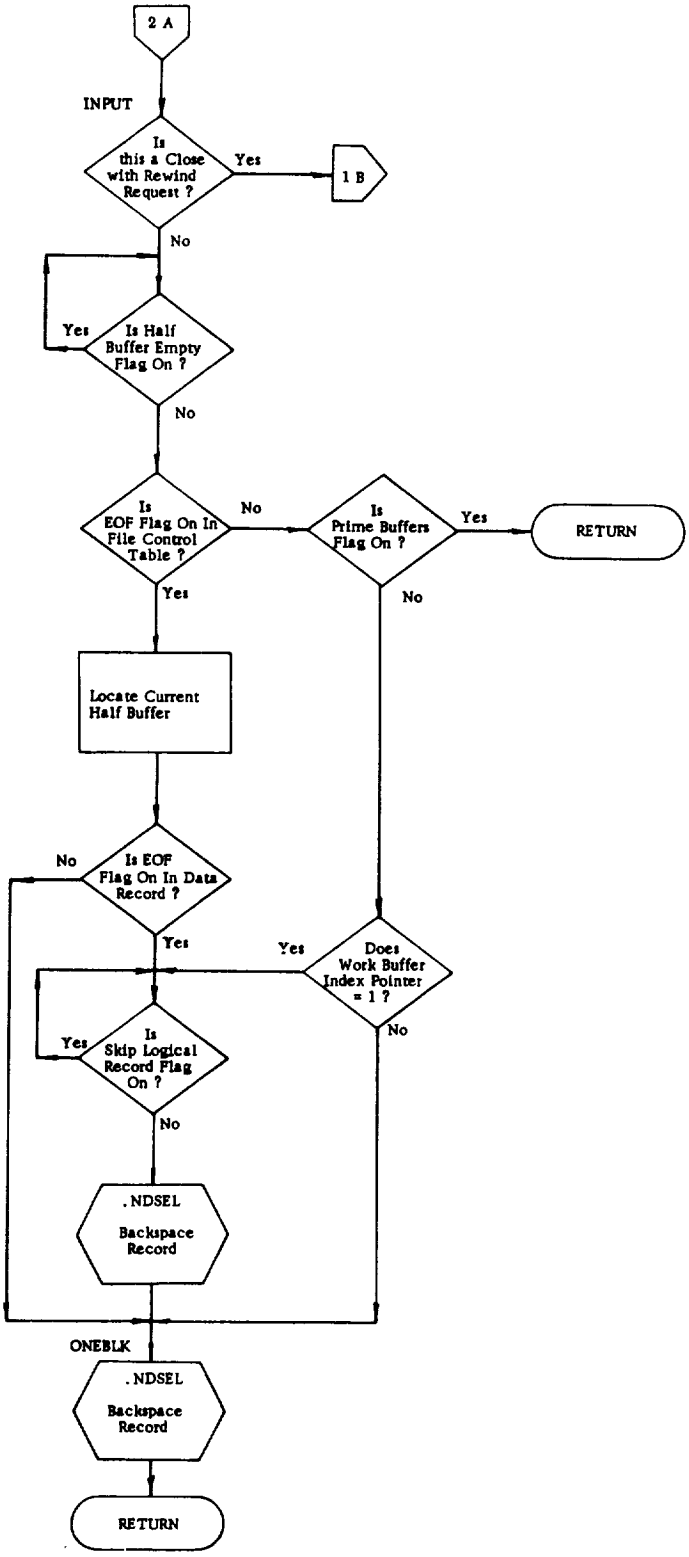


Figure 3(b). Flowchart for the DCS assembly language GINØ routine CLØSE - deck name CLØSED.

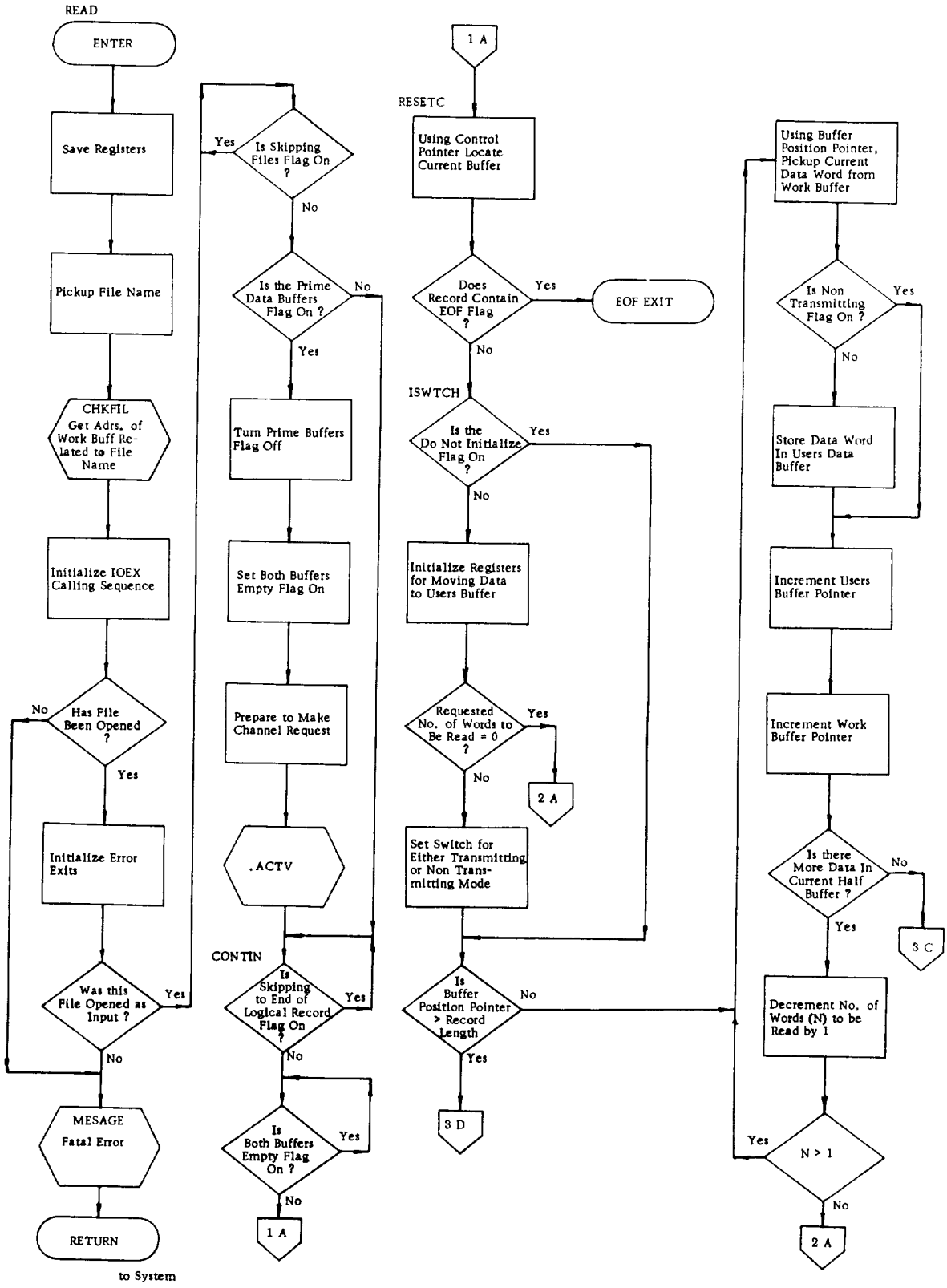


Figure 4(a). Flowchart for the DCS assembly language GINØ routine READ - deck name READD.

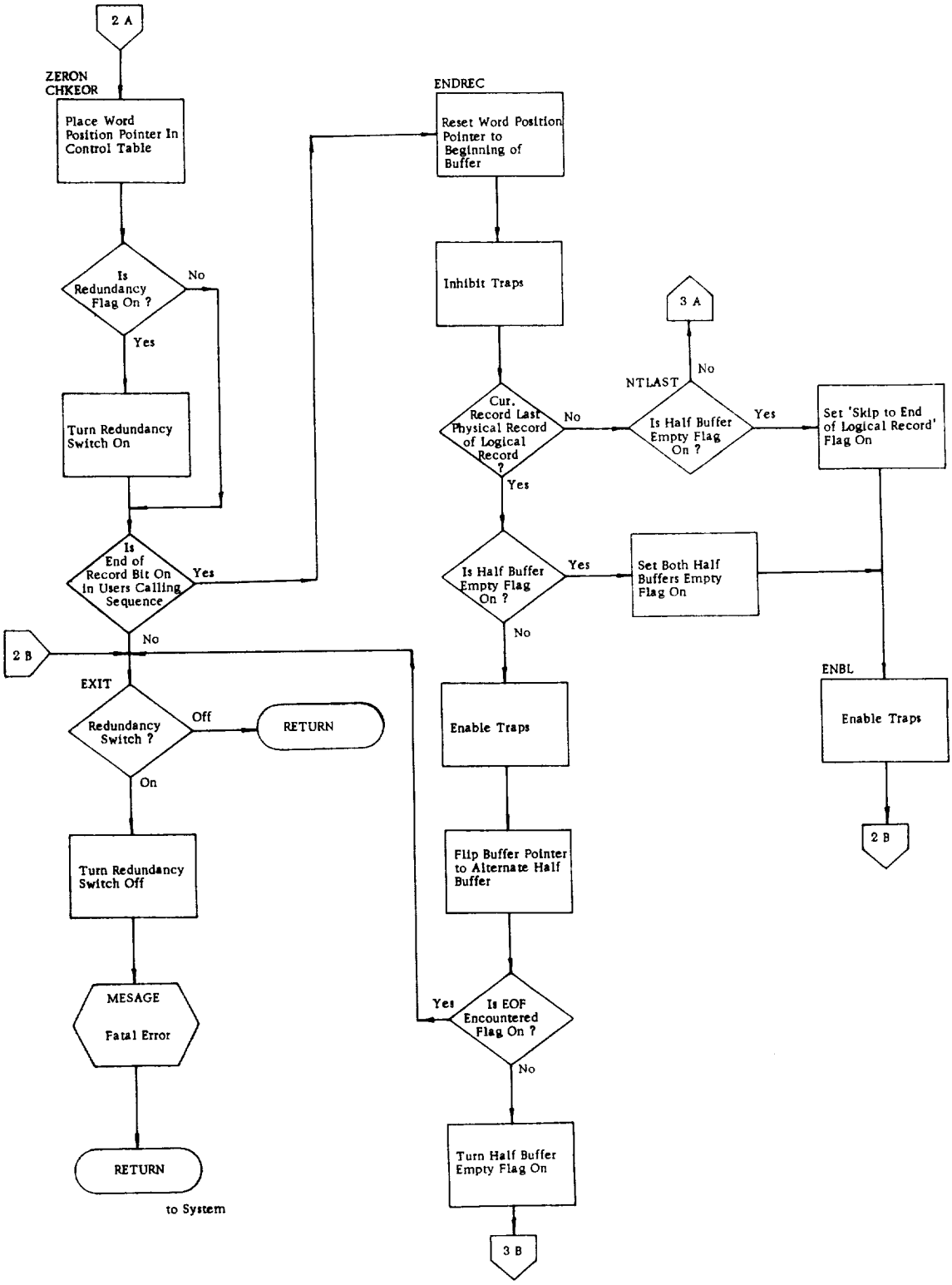


Figure 4(b). Flowchart for the DCS assembly language GINØ routine READ - deck name READD.

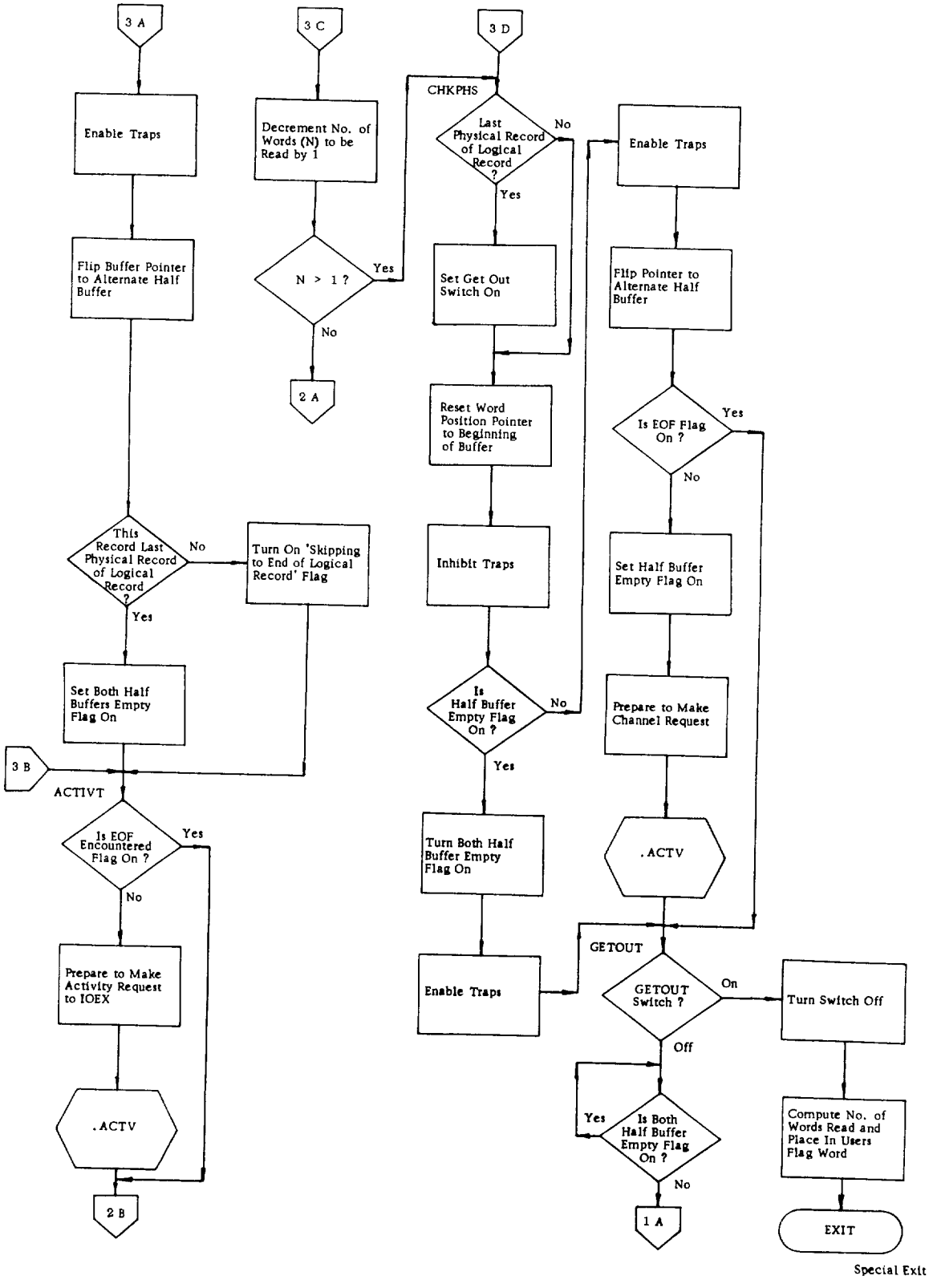


Figure 4(c). Flowchart for the DCS assembly language GINØ routine READ - deck name READD.

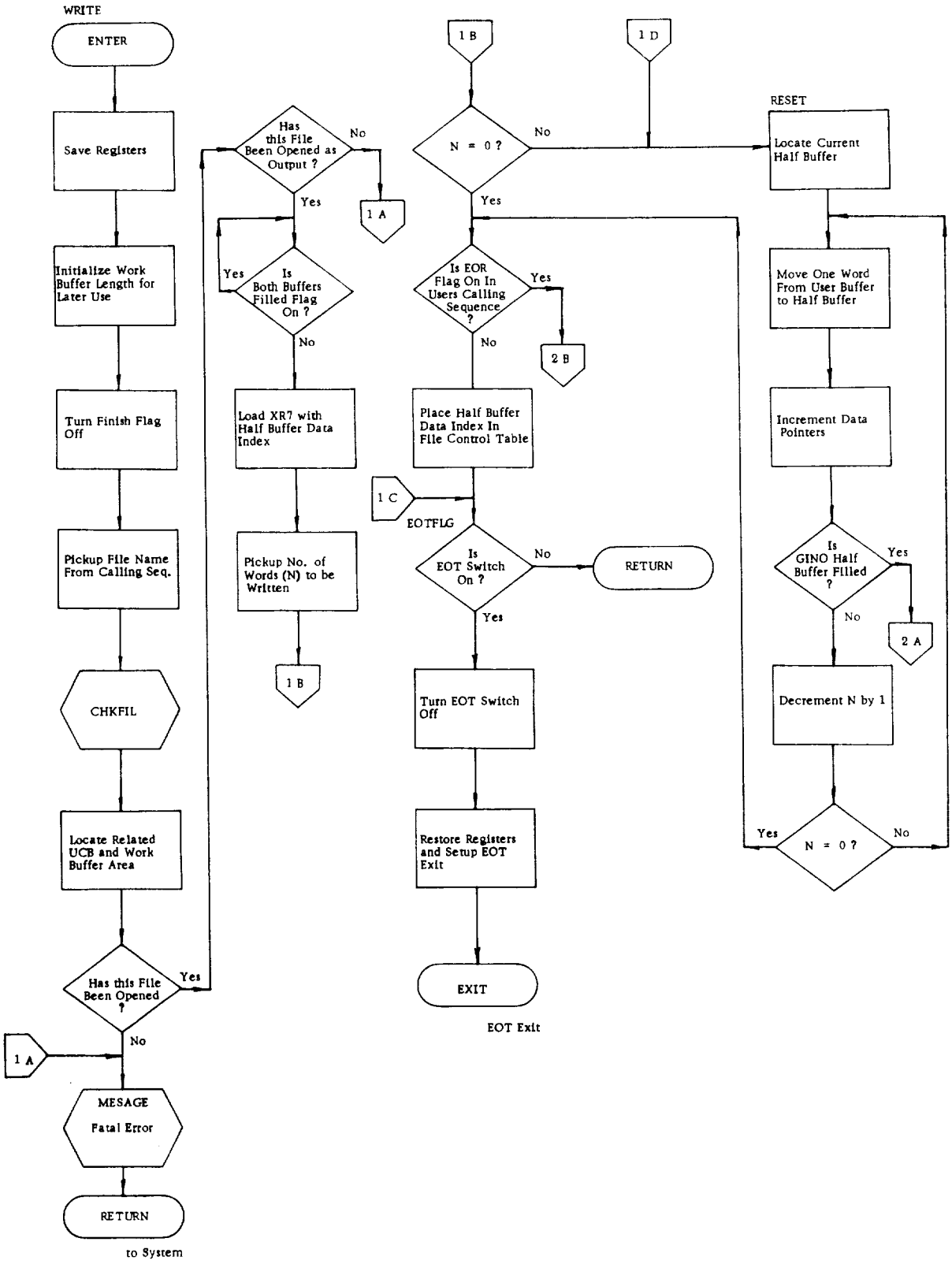


Figure 5(a). Flowchart for the DCS assembly language GINØ routine WRITE - deck name WRITEN .

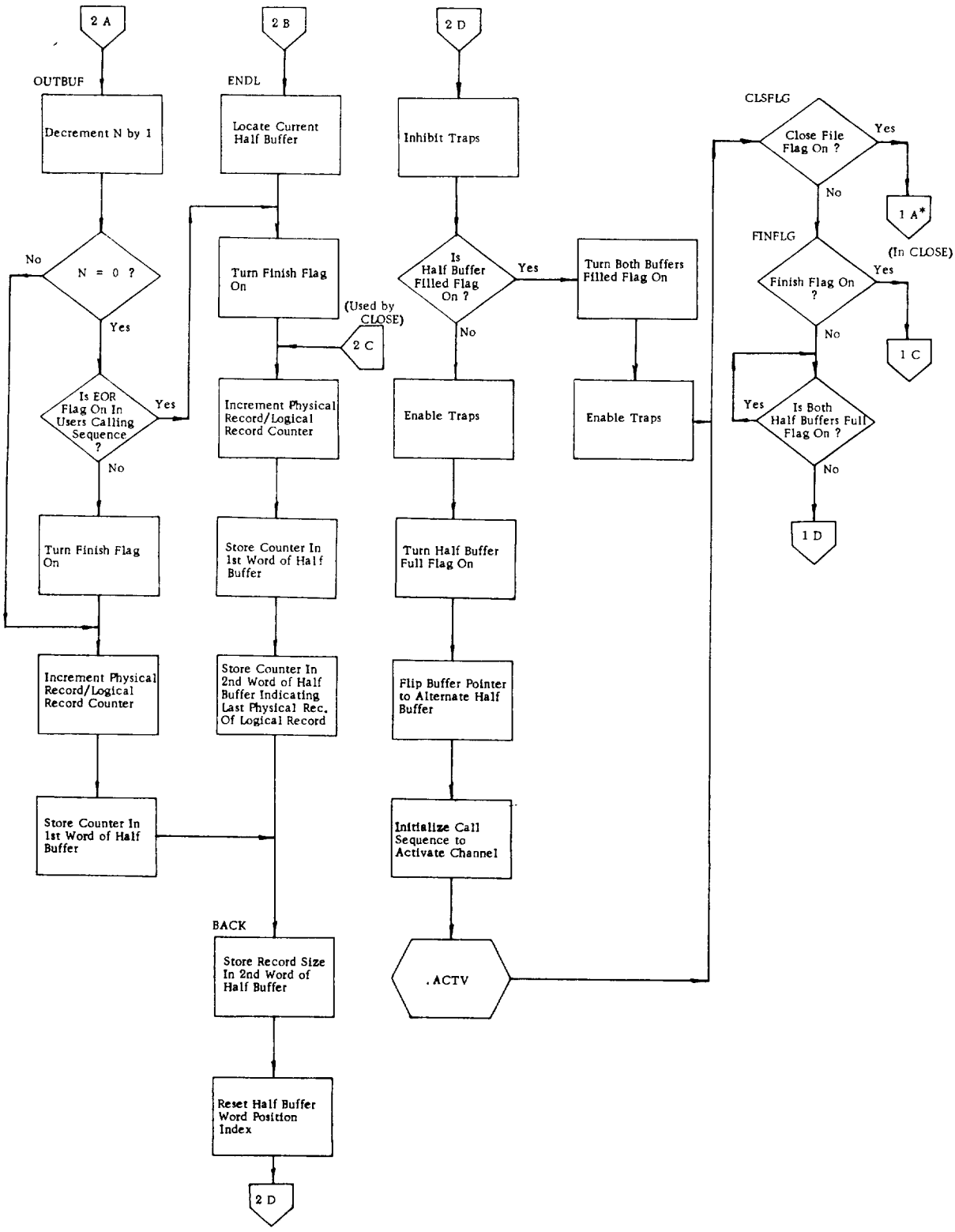


Figure 5(b). Flowchart for the DCS assembly language GINØ routine WRITE - deck name WRITEN.

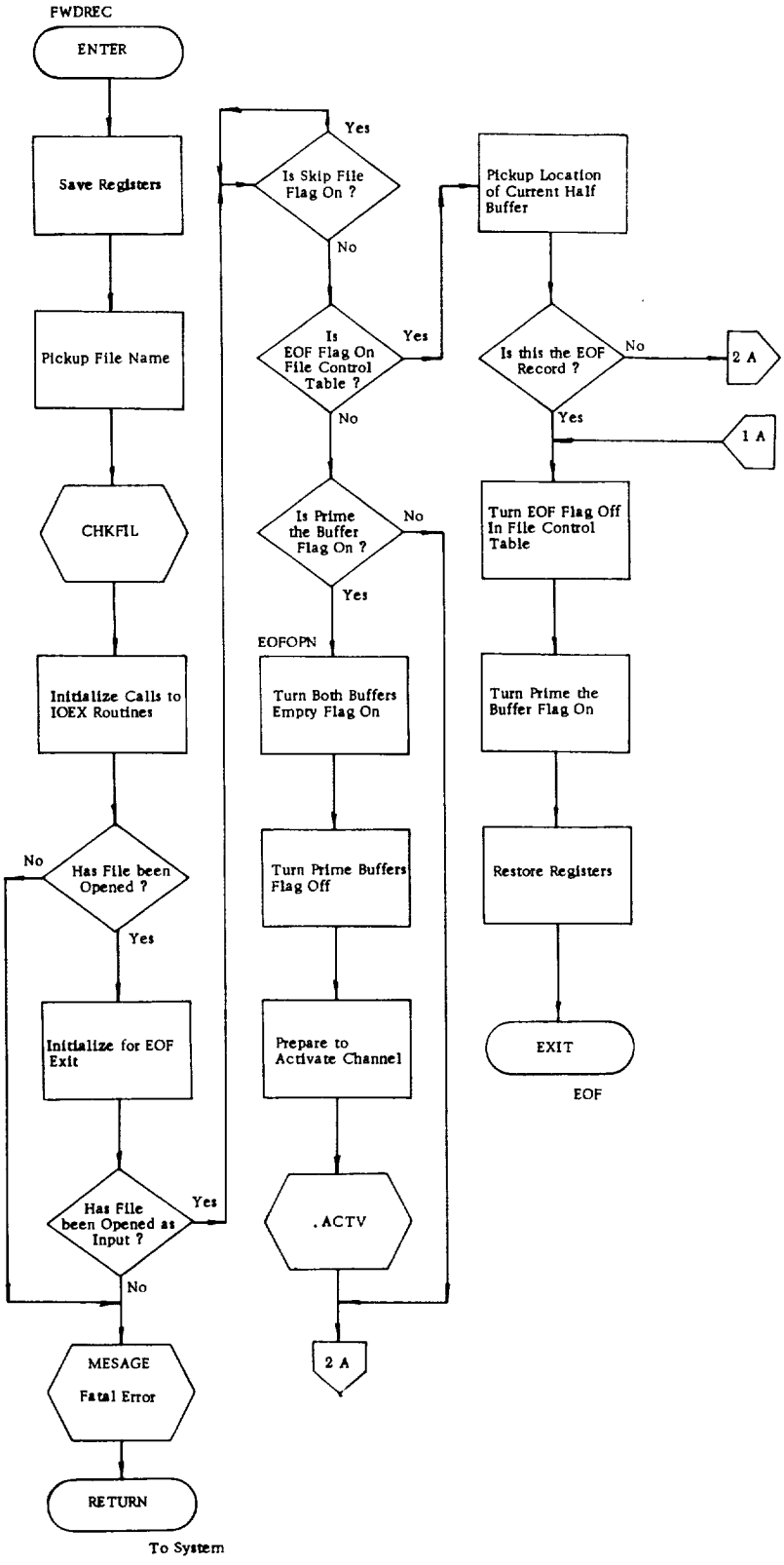


Figure 6(a). Flowchart for the DCS assembly language GINØ routine FWDREC - deck name FØRWRD.

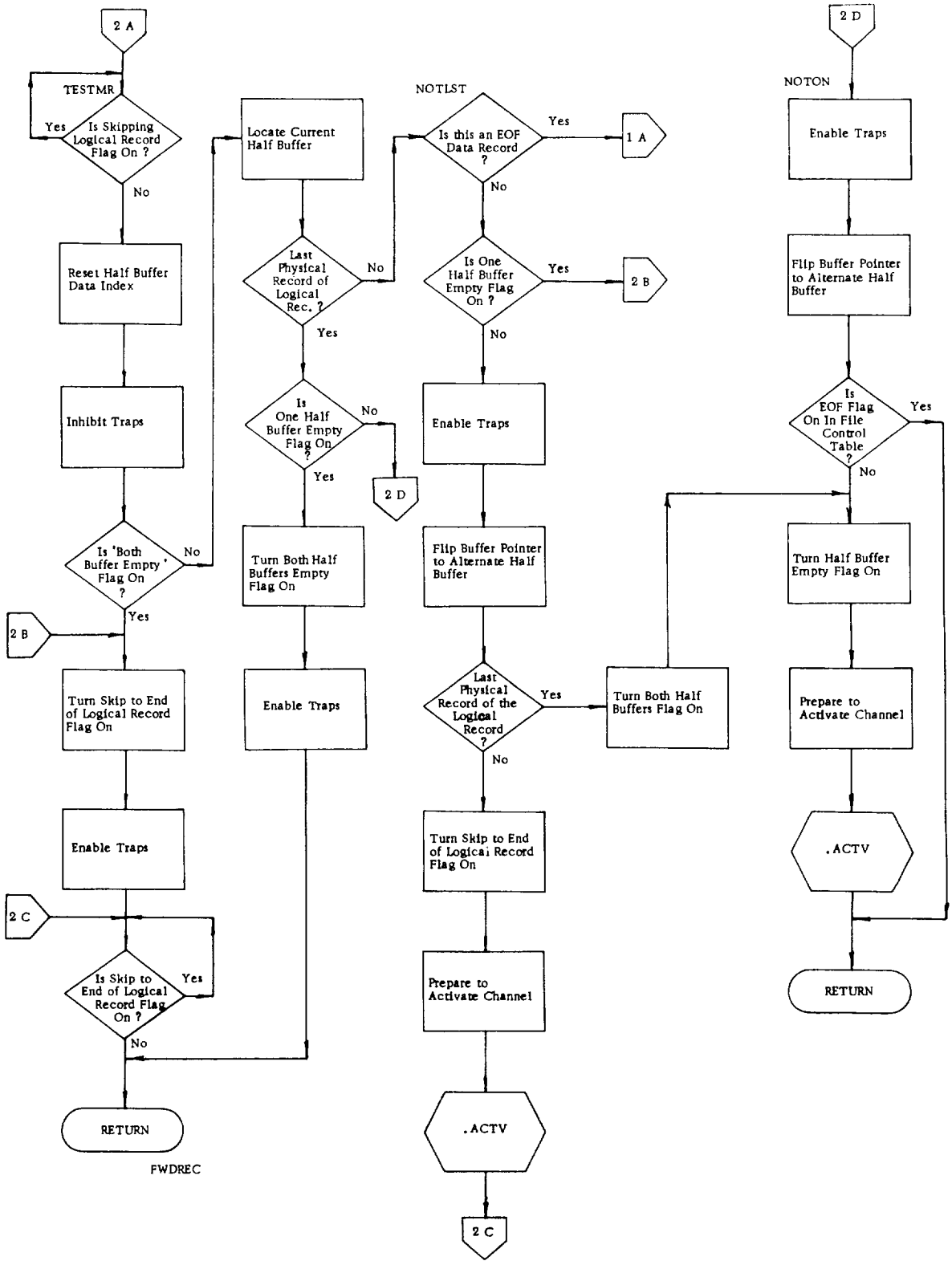


Figure 6(b). Flowchart for the DCS assembly language GINØ routine FWDREC - deck name FØRWRD.

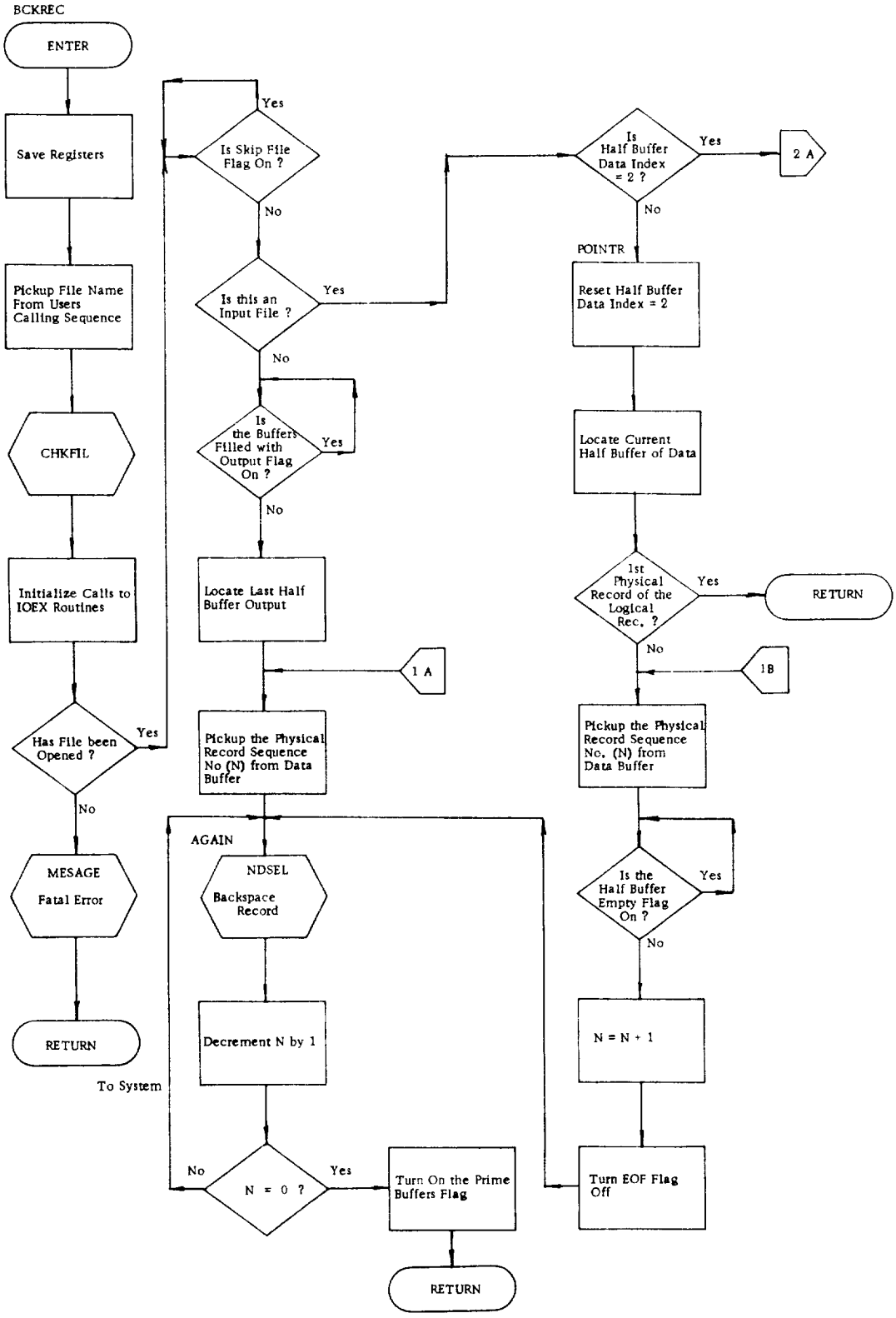


Figure 7(a). Flowchart for the DCS assembly language GINØ routine BCKREC - deck name BKREC.

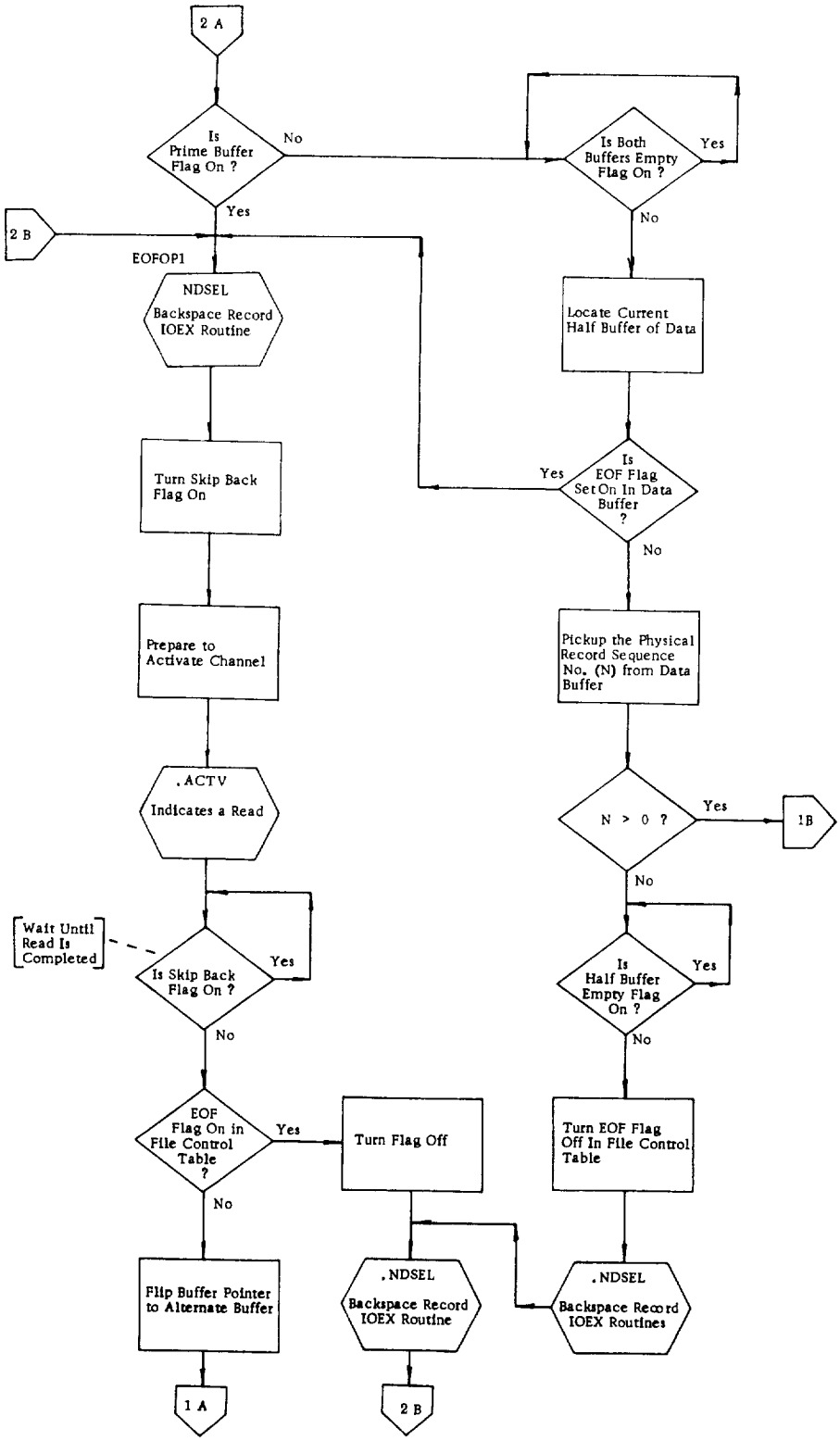


Figure 7(b). Flowchart for the DCS assembly language GINØ routine BCKREC - deck name BKREC.

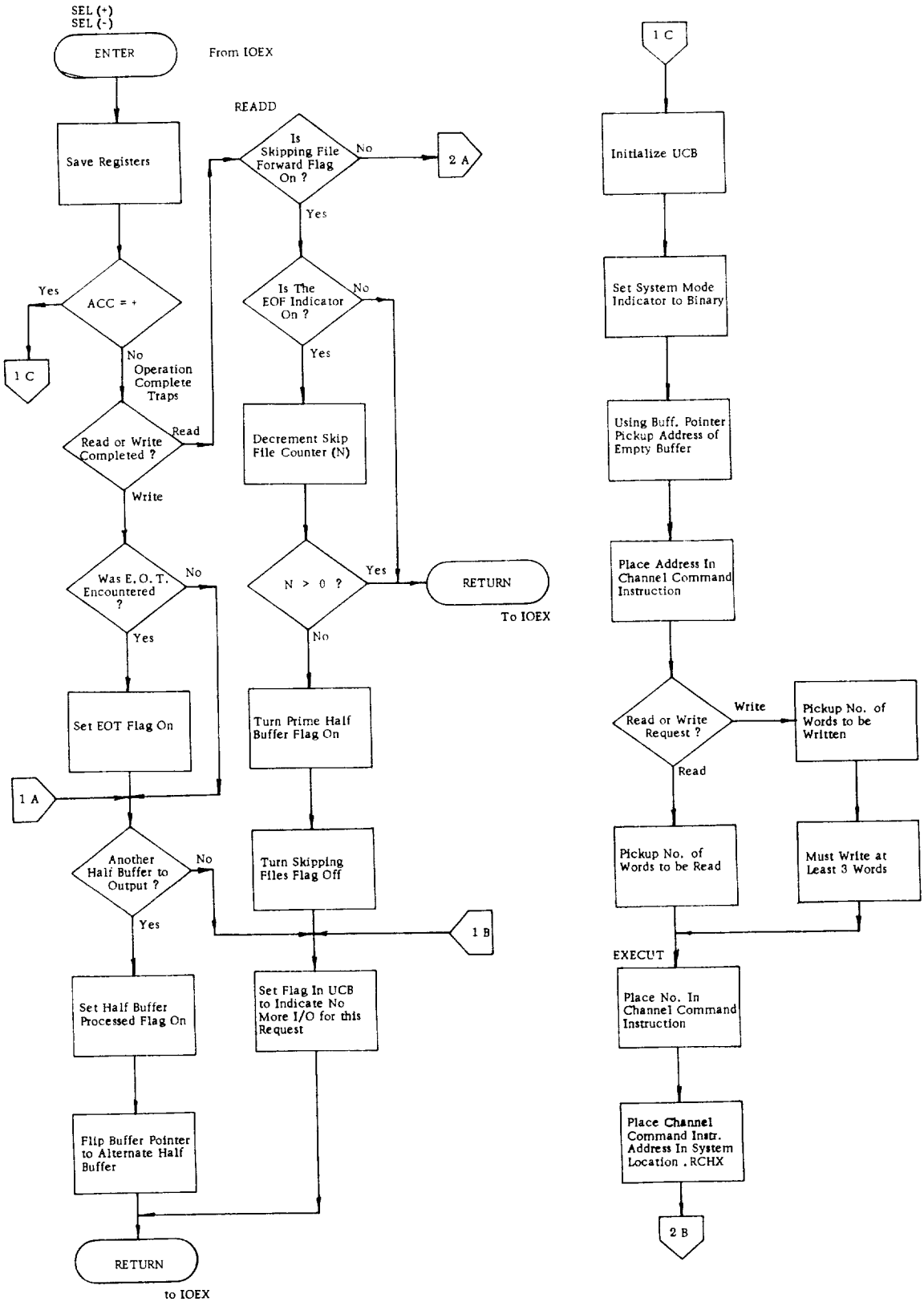


Figure 8(a). Flowchart for the DCS assembly language GINØ routine SEL+ - deck name SELECT.

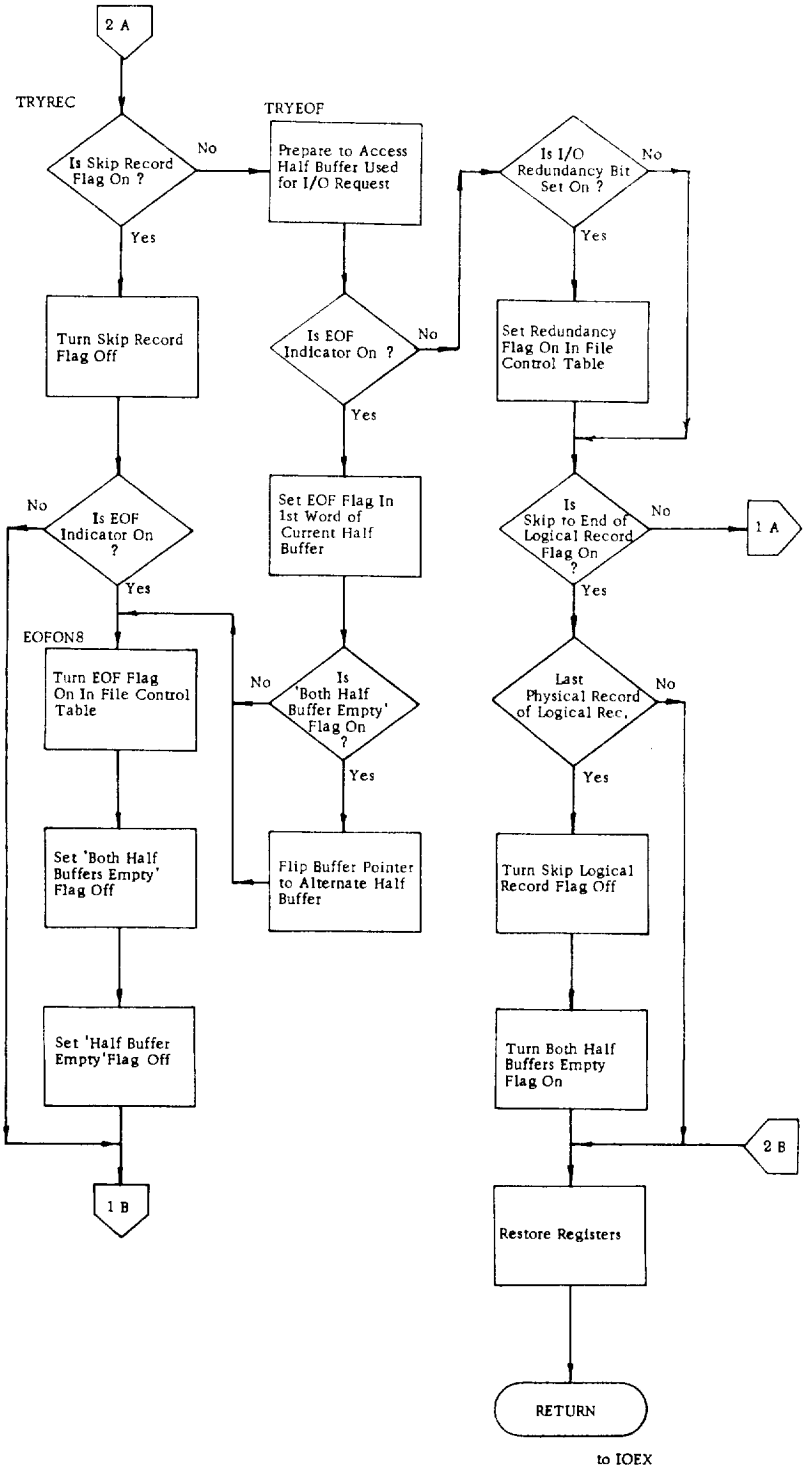


Figure 8(b). Flowchart for the DCS assembly language GINØ routine SEL+ - deck name SELECT.

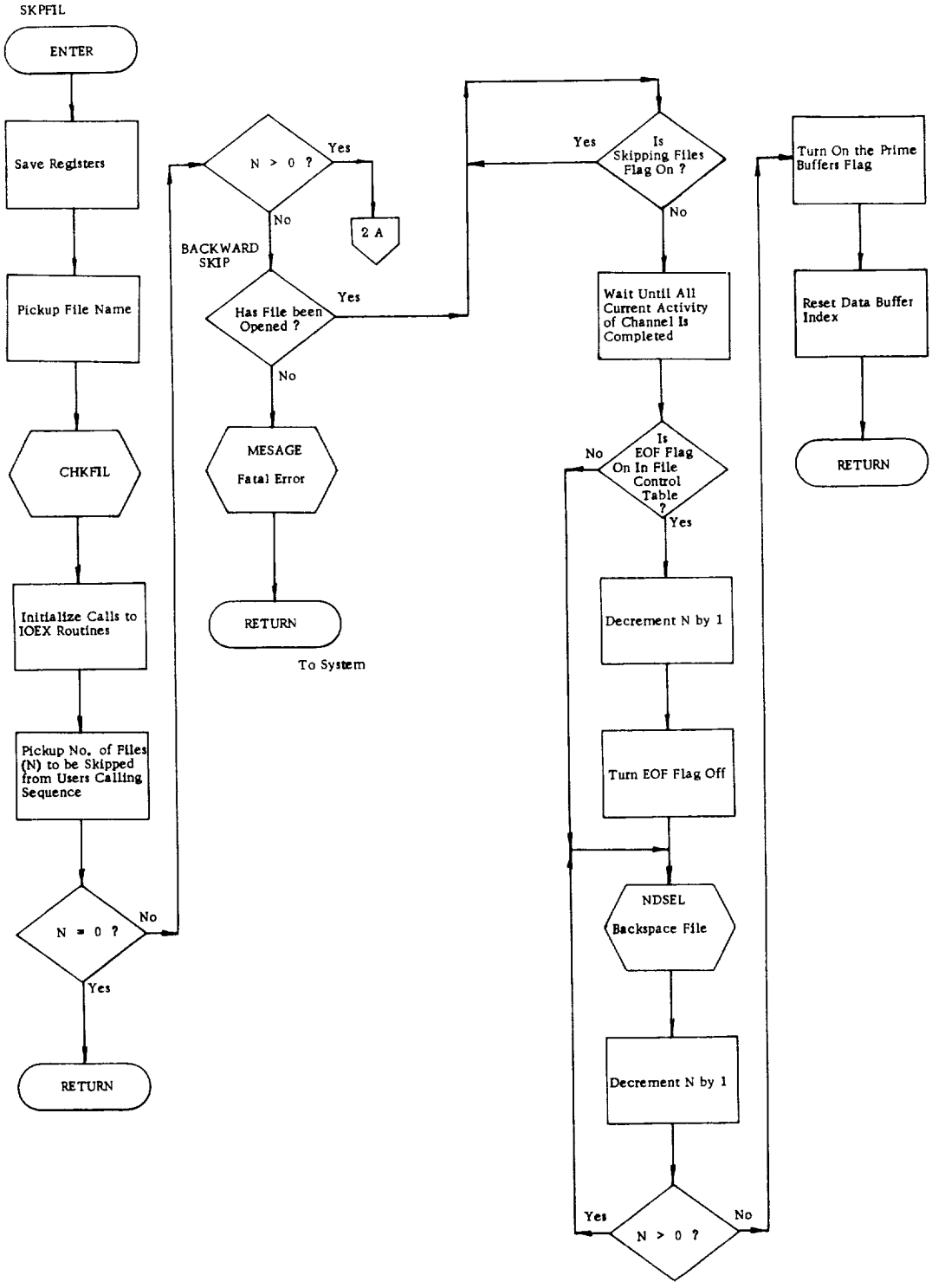


Figure 9(a). Flowchart for the DCS assembly language GINØ routine SKPFIL - deck name SKIPF.

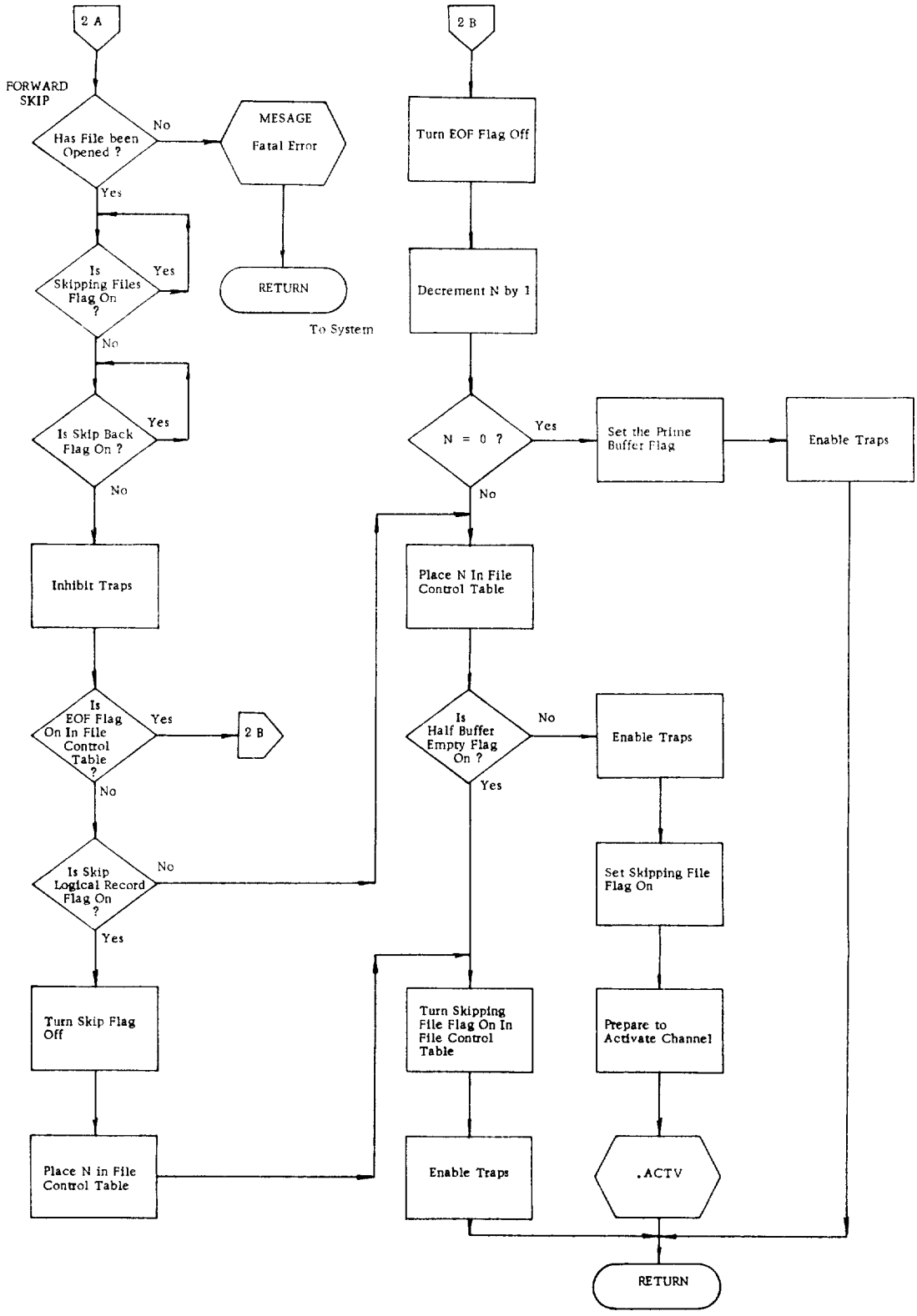


Figure 9(b). Flowchart for the DCS assembly language GINØ routine SKPFIL - deck name SKIPF.

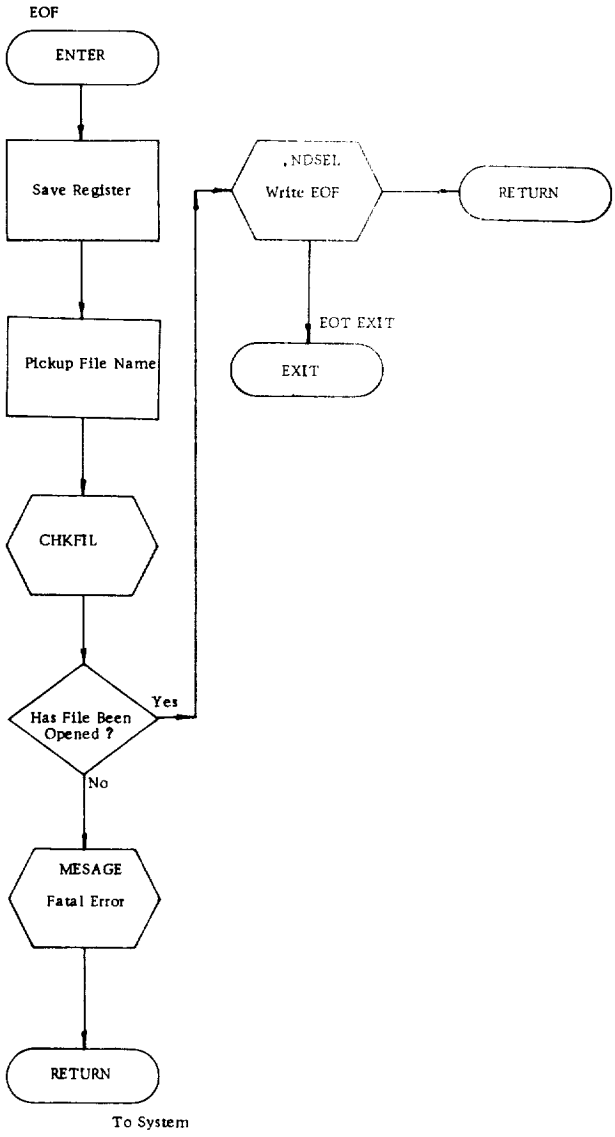


Figure 10. Flowchart for the DCS assembly language GINØ routine EØF - deck name MISC .

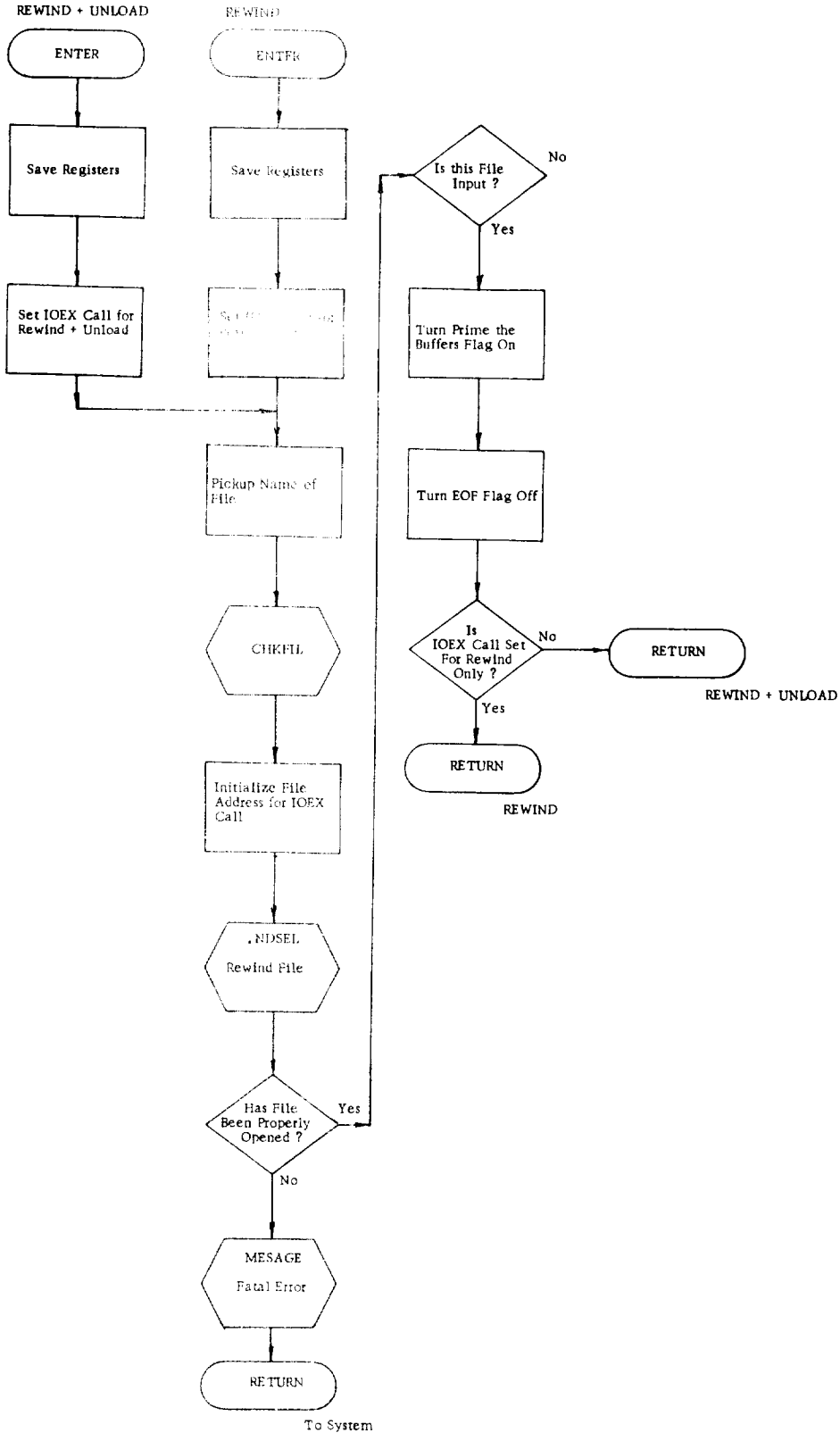
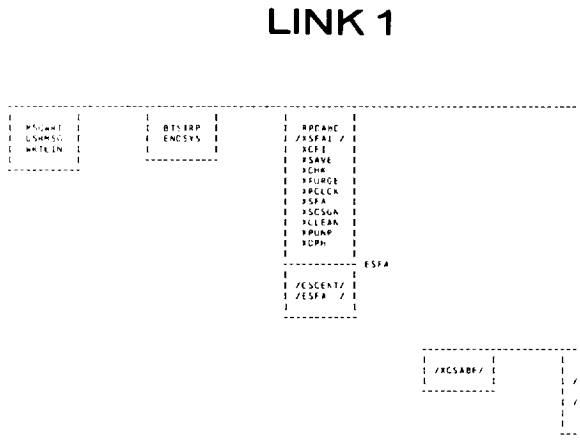


Figure 11. Flowchart for the DCS assembly language GINØ routine REWIND - deck name MISC .

5.2.9 DCS Overlay Diagrams

Figures 12, 13, 14, 15 and 16 on the following fold-out pages represent the NASTRAN overlay structure for the DCS. Since the DCS overlay structure was duplicated as nearly as possible on the other machines, these diagrams can be used to follow the exact overlay implementation on these machines. Note that link 8 does not appear in the following diagrams. This link can be used for the addition of a new link (see section 6.11).



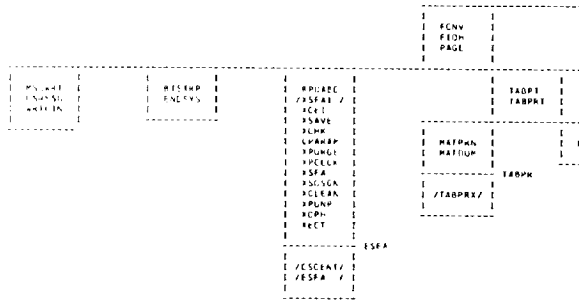
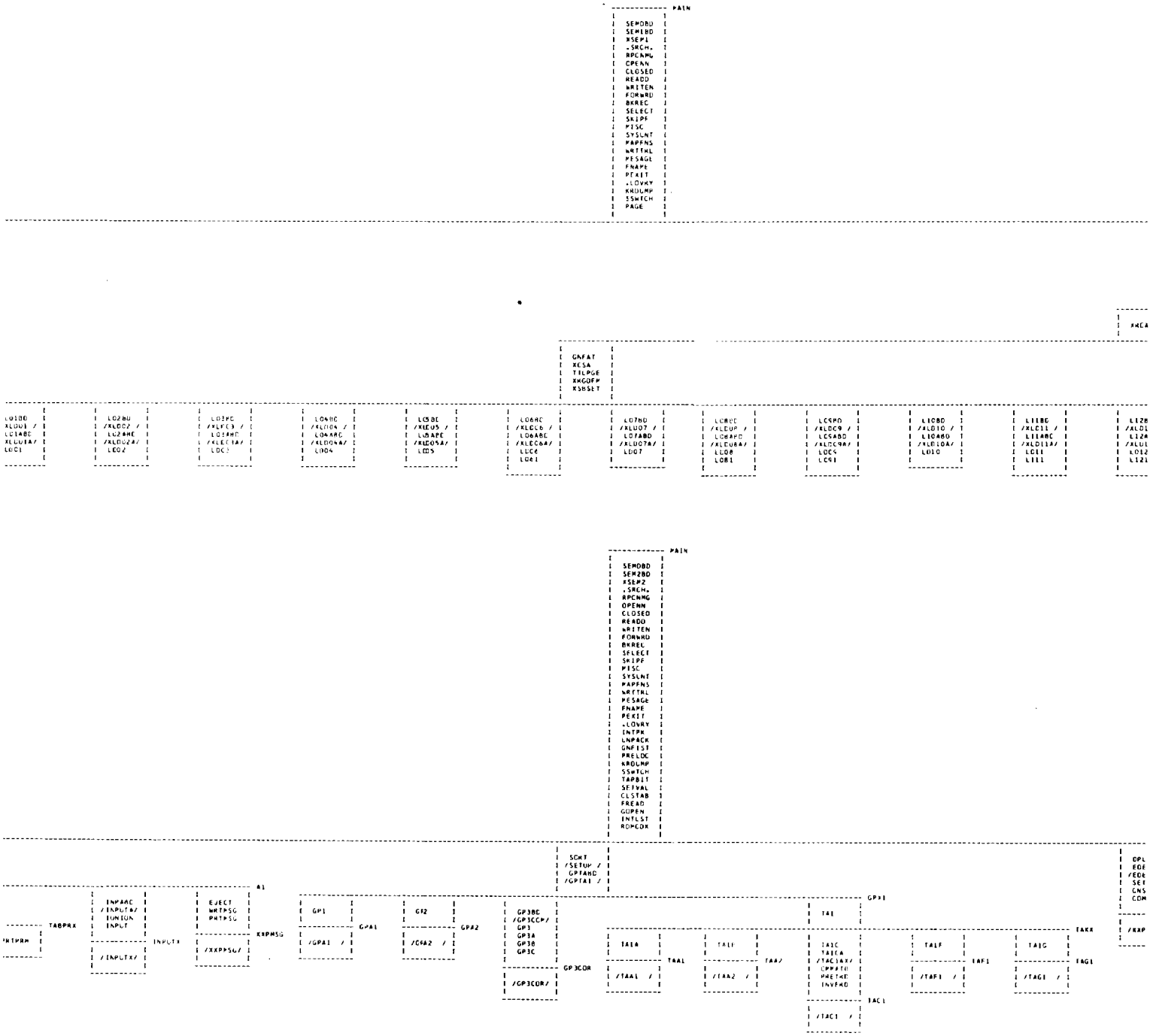


FIG. 12 - OVERLAY STRUCTURES FOR

- links 1 and 2 on the IBM 7094/7040(44) DCS.


MAIN
SEMOBD
SEM1BD
RSEP1
.SRCH.
RPCNMG
OPENN
CLOSED
READD
WRITEN
FORWRD
BKREC
SELECT
SKIPF
MISC
SYSLNT
MAPFNS
WRTTKL
MESAGE
FNAME
PEKIT
.LOVRY
KRDUMP
SSWTCH
PAGE
RXCA
GNFAT
XCSA
TILPGE
XKGOFM
XSBSET
LO1DD
XLDD1
LO14BD
XLDD1A
LOC1
LO2BD
/XLDD2
LO24HE
/XLDD2A
LOC2
LO3HC
/XLEC3
LO34HD
/XLEC3A
LOC3
LO4BC
/XLD04
LO44BC
/XLDD4A
LOC4
LO5BE
/XLEC5
LO54BE
/XLDD5A
LOC5
LO6BC
/XLEC6
LO64BE
/XLEC6A
LOC6
LO61
LO7BD
/XLD07
LO74BD
/XLDD7A
LOC7
LO8HC
/XLEC8
LO84HD
/XLDD8A
LOC8
LO81
LC9MD
/XLDC9
LC94BD
/XLDC9A
LOC9
LC91
L10BD
/XLD10
L104BD
/XLD10A
LO10
L11BC
/XLC11
L114BC
/XLD11A
LO11
L111
L12B
/XLD1
L124
/XLUL
LO12
L121

MAIN
SEMOBD
SEM2BD
RSEP2
.SRCH.
RPCNMG
OPENN
CLOSED
READD
WRITEN
FORWRD
BKREC
SELECT
SKIPF
MISC
SYSLNT
MAPFNS
WRTTKL
MESAGE
FNAME
PEKIT
.LOVRY
INTPK
LNPACK
GNFIST
PRELOC
KRDUMP
SSWTCH
TAPBIT
SETVAL
CLSTAB
FREAD
GOPEN
INTLST
ROMCOR
SCRT
/SETUP /
GPTABD
/GPTAI /
OPL
EOF
/EOE
SER
CNS
COM
/RXP
A1
TABPRX
INTPRM
ENPARC
/INPUTA/
TUNTOK
INPUT
INPUTX
/INPUTX/
EJECT
WRTMSG
PRTMSG
RXPMSG
/RXPPSG/
GP1
/GPA1 /
GPA1
GP2
/GA2 /
GPA2
GP3BE
/GP3CCP/
GP3
GP3A
GP3B
GP3C
/GP3CDR/
GP3COR
TAI
GP31
TAIA
/TAAL /
TAAL
TAIF
/TAA2 /
TAA2
TAIC
TAICA
/TAC1AX/
CPPATH
PRETHC
INVEAD
/TAC1 /
TAC1
TAKA
TAIF
/TAF1 /
TAF1
TAIG
/TAG1 /
TAG1

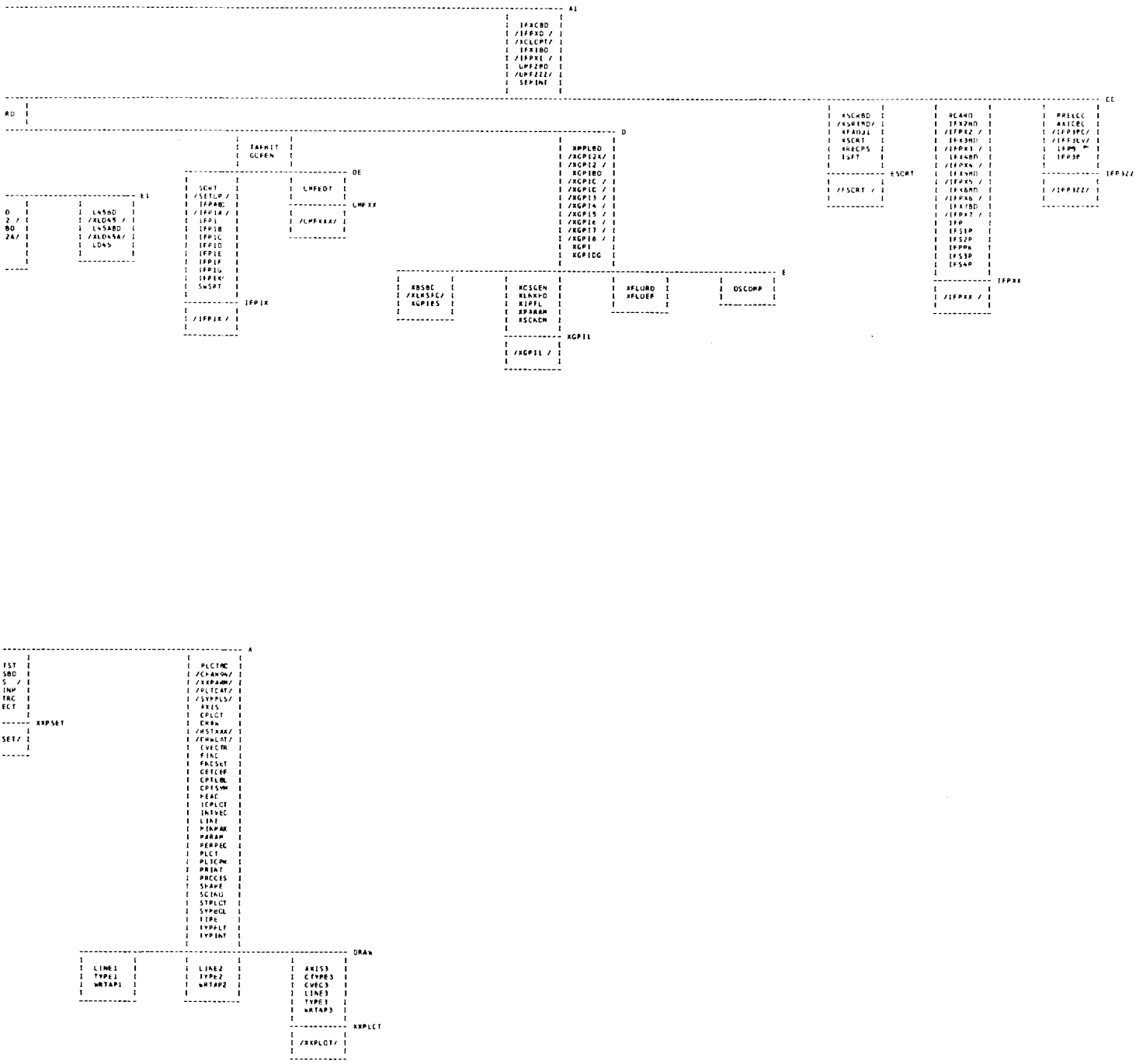
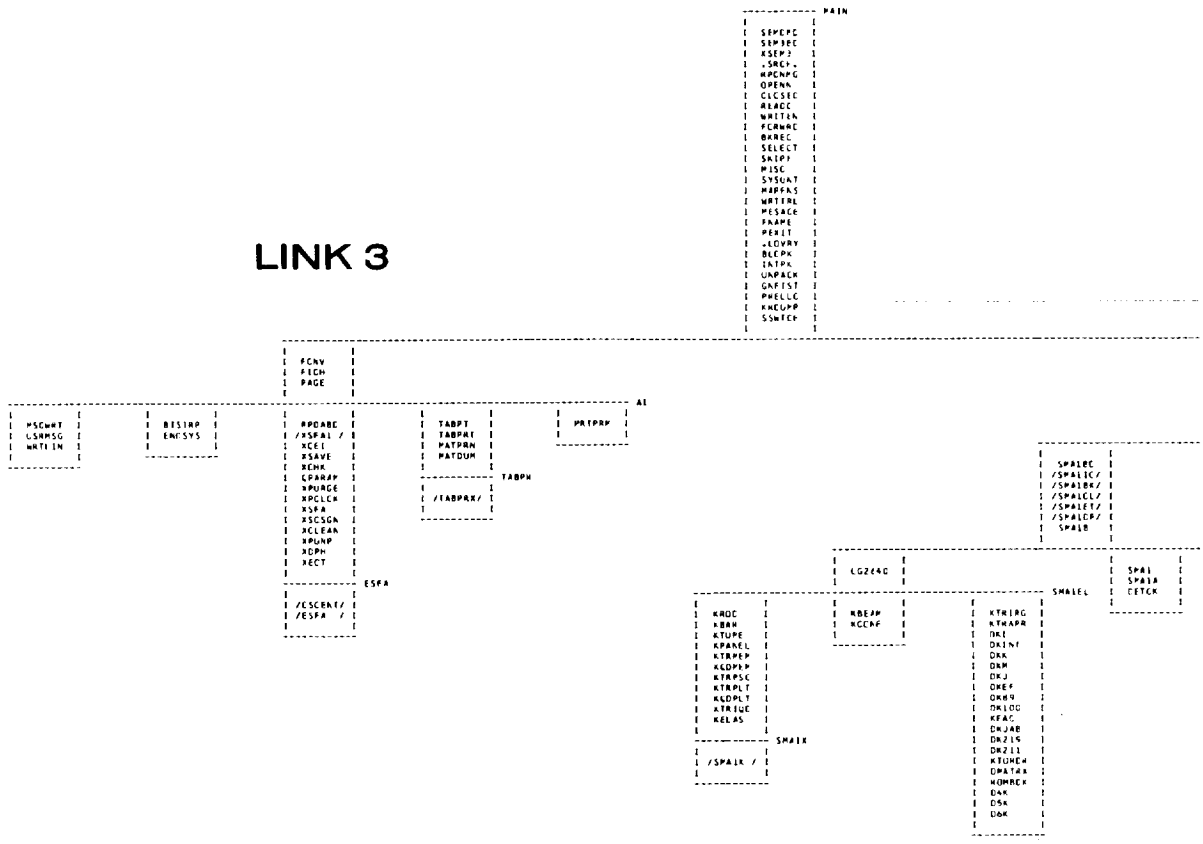


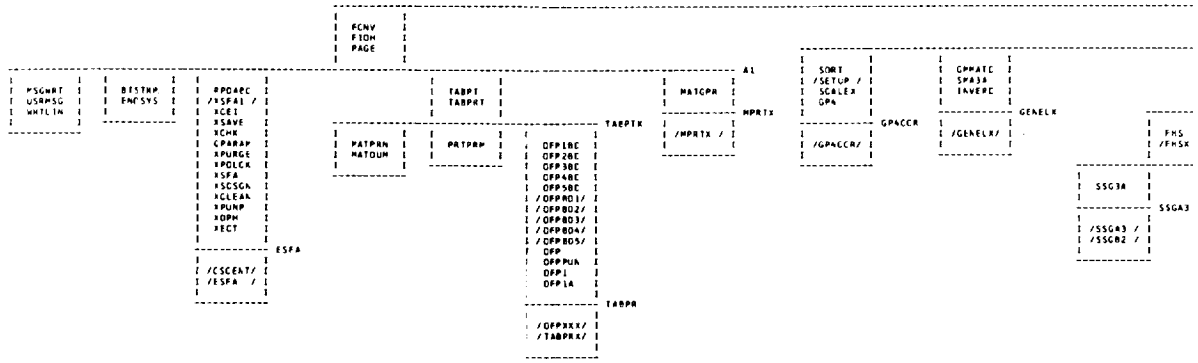
Figure 12. Overlay structure for links 1 and 2 on the IBM 7094/7040(44) DCS.

Figure 13. Overlay structure for links 3 and 4 on the IBM 7094/7040(44)

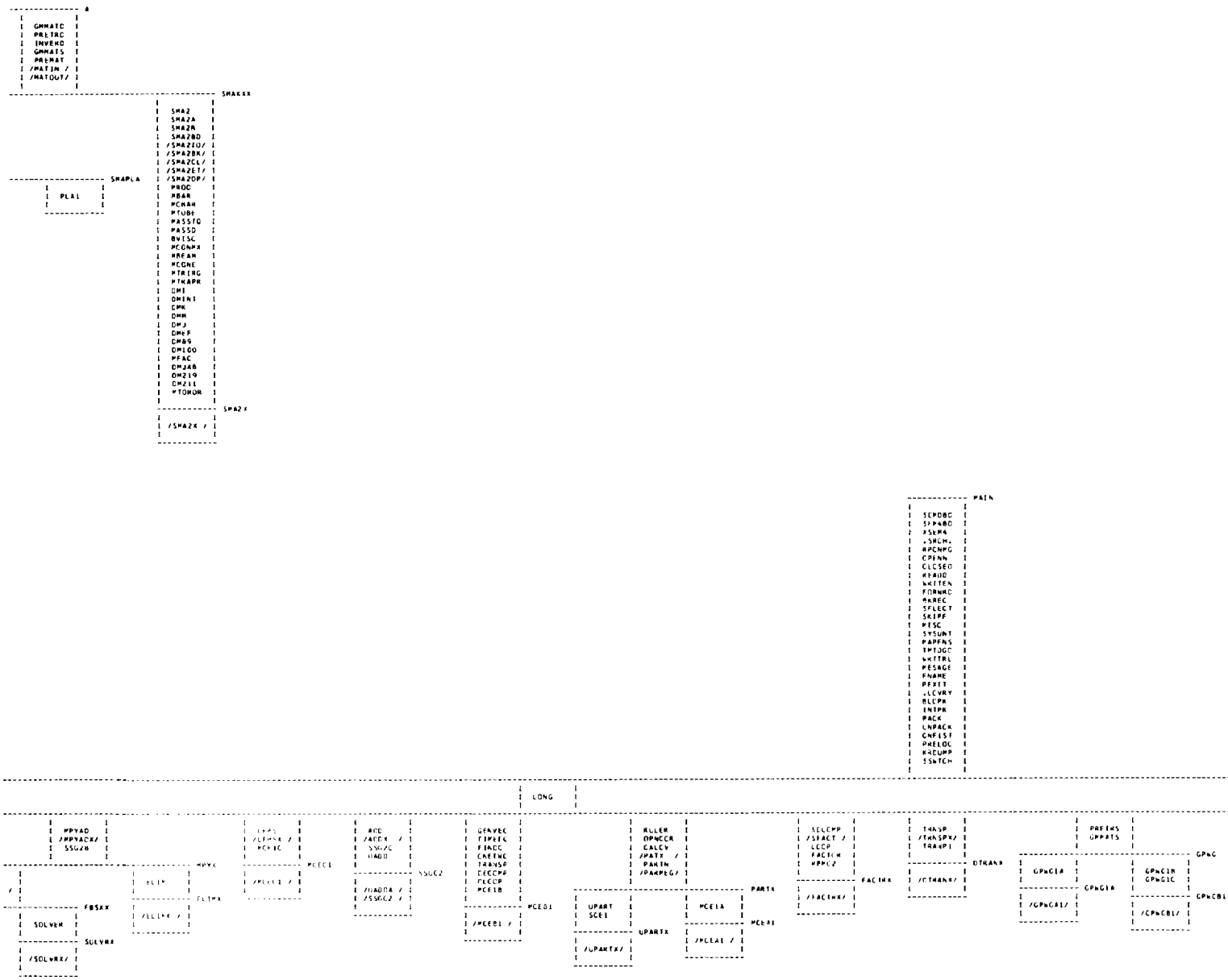
LINK 3



LINK 4



DCS.


A
GMMATC
PRETRC
INVERO
GMMATS
PREMAT
/MATIN /
/MATOUT/
SMAKXX
SMA2
SMA2A
SMA2A
SMA2BD
/SMA2IO/
/SMA2BK/
/SMA2CL/
/SMA2ET/
/SMA2OP/
PROC
MBAR
MCMAH
MTUBF
MASSTQ
MASSD
BVESC
MCGMMX
MREAM
MCGNE
MTRING
MTKAPR
DMI
DMINT
CMK
DMM
DMJ
DMEF
DM89
DM100
MFAC
DMJA8
DM219
DMJ11
MTORQR
SMA2X
/SMA2X /
SMAPLA
PLAI
MAIN
SEPDBG
SPP480
XSEM4
.SKGM.
MPCNPG
CPENN
CLCSEO
MPRID
WRITEN
FORMAC
MAREC
SFLECT
SKIPF
MESC
SYSUNT
MAPENS
TMTOGC
WKTTRL
MESAGE
FNAME
PEXIT
.LCVRY
BLCPR
INTPR
PACK
UNPACK
GNFIST
MADUMP
SSWTCH
LONG
MPYAD
/MPYACRX/
SSGJ8
MPYX
/ELIMX /
ELIMX
/ELIMX /
ELIMX
/MPCEE1 /
MCEE1
ACD
/ACDRX /
SSGJC
HADD
/HADRA /
/SSGC2 /
SSGC2
GENVEC
TIMREC
FIACC
CNETWC
TRANSP
DECCMP
MCCOP
MCEIB
/MCEB1 /
MCEB1
RLLER
OPNECR
CALCV
/PATX /
PAMTN
/PAMPEG/
/UPARTX/
UPARTX
MCE1A
/MCEAE /
MCEAI
PARTX
SELCMP
/SRACT /
LCCP
FACTCM
MPMC2
/FACTMX/
FACTMX
TRNSP
/TRNSPX/
TRANPI
/DTRANX/
DTRANX
PREFMS
GPPATS
GPnC
GPnC1A
/GPnCA1/
GPnC1A
GPnC1B
GPnC1C
GPnCB1
/GPnCB1/
SOLVER
FBSXX
/SOLVRX/
SOLVRX

5.2-33C.

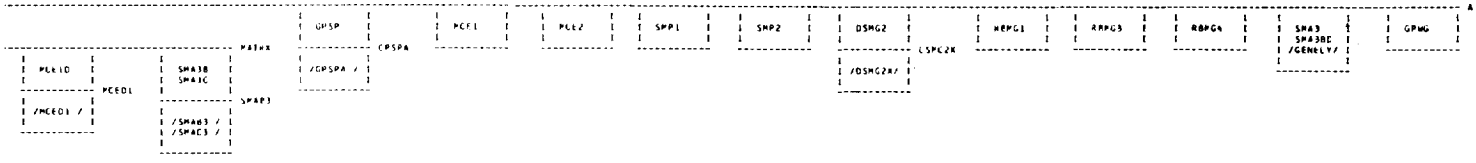
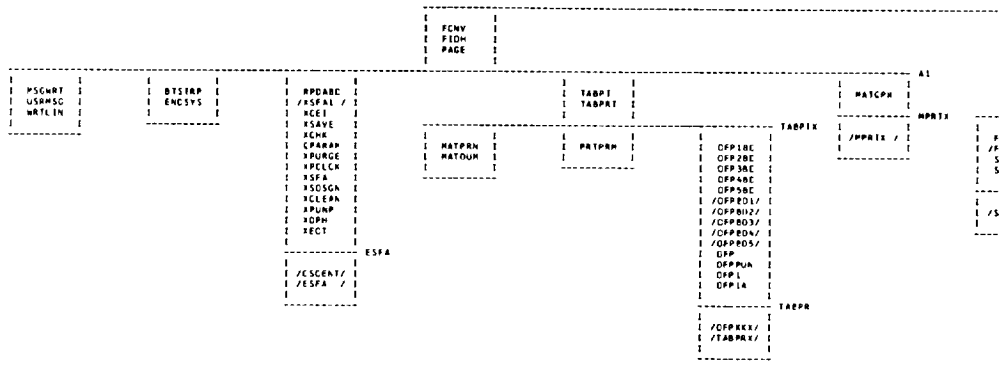


Figure 13. Overlay structure for links 3 and 4 on the IBM 7094/7040(44) DCS.

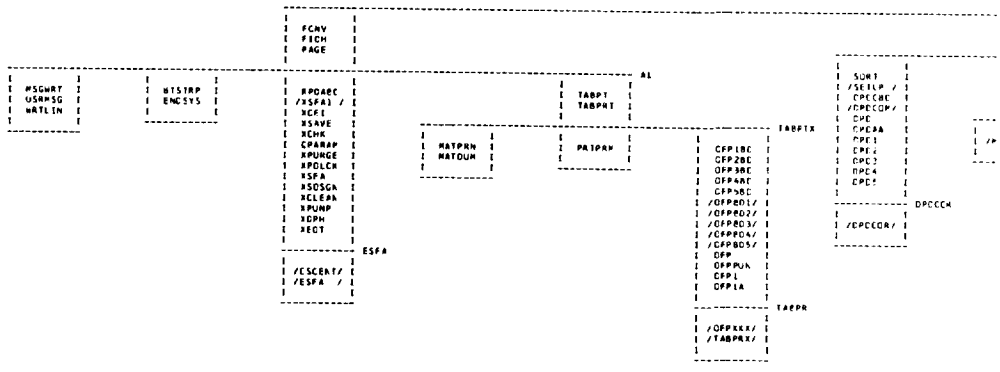
Figure 14. Overlay structure for links 5 and 6 on the IBM 7

)

LINK 5



LINK 6



094/7040(44) DCS.

5.2-35B

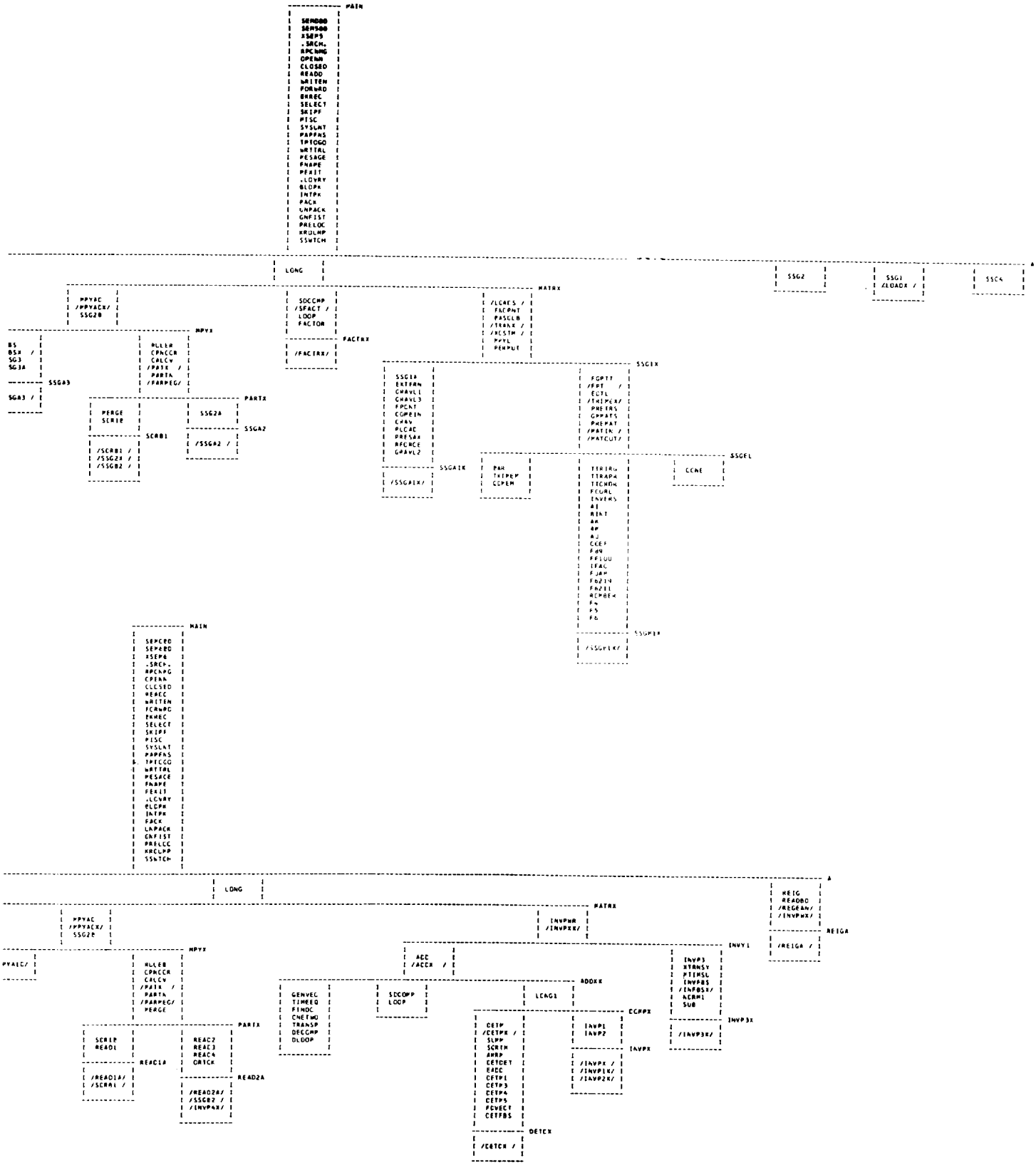
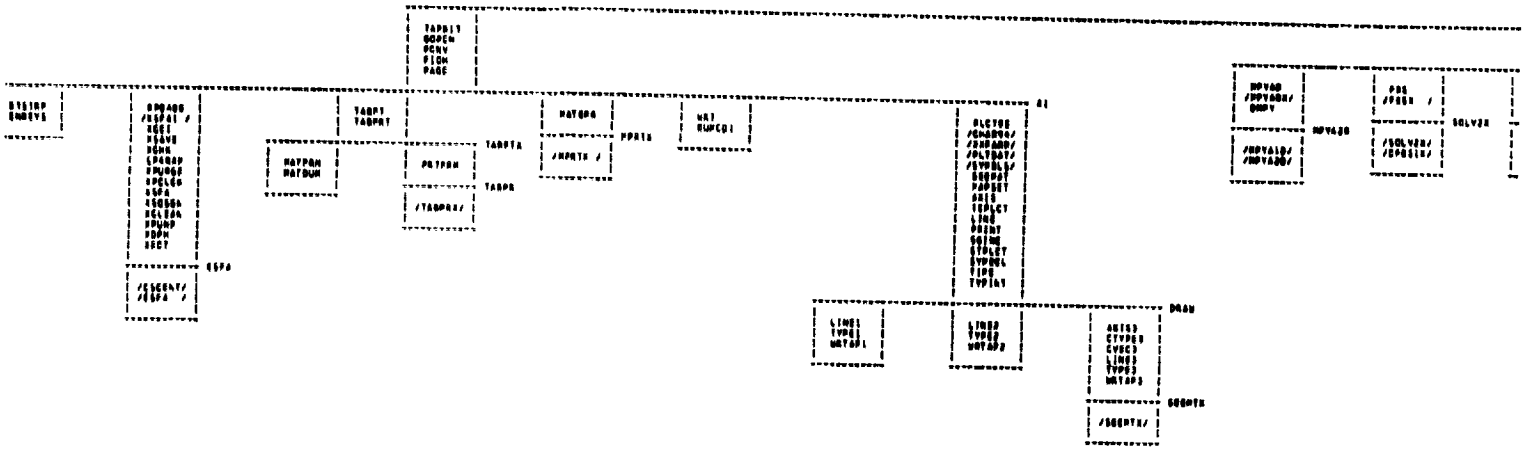


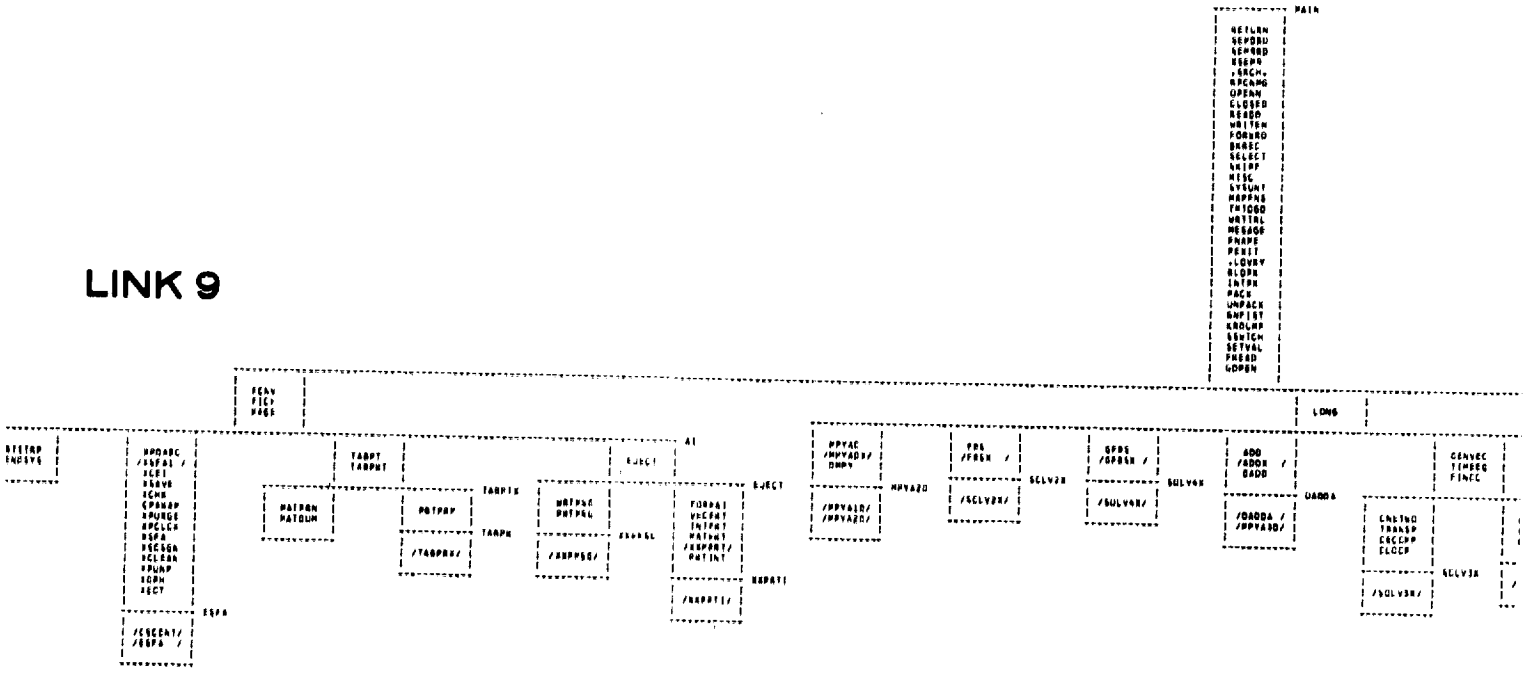
Figure 14. Overlay structure for links 5 and 6 on the IBM 7094/7040(44) DCS.

FIG.
15. Overlay structure for links 7 and 9 on the IBM 7094/7040(44) DCS.

LINK 7



LINK 9



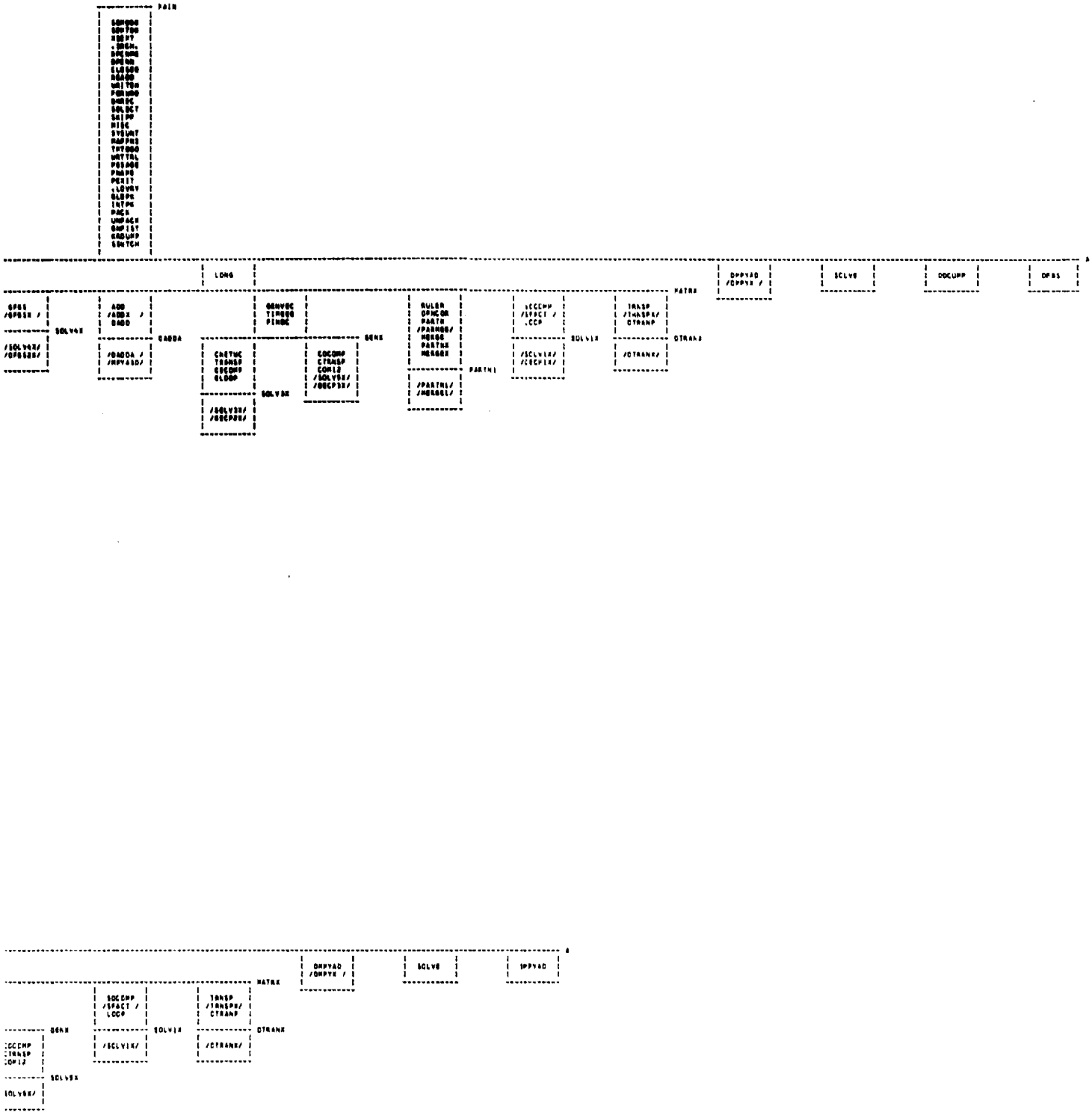
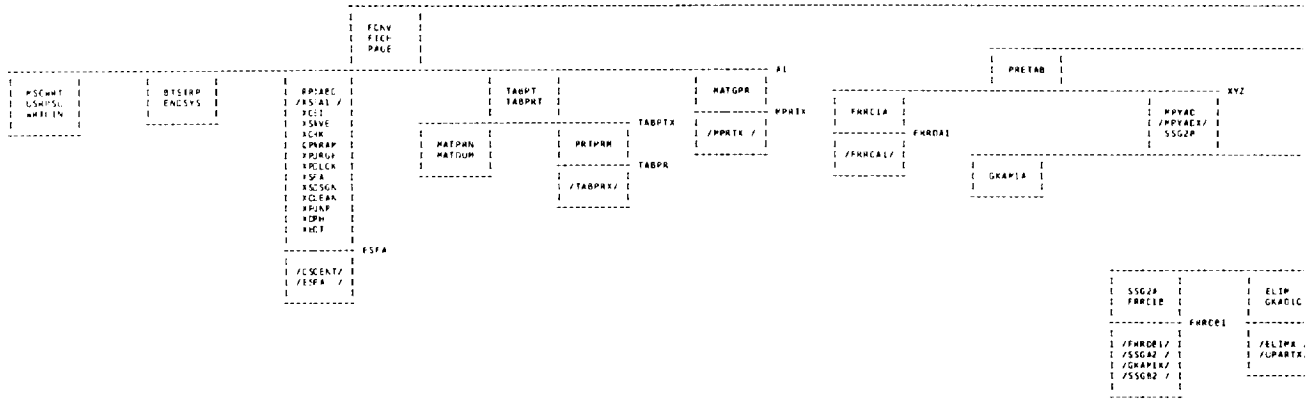


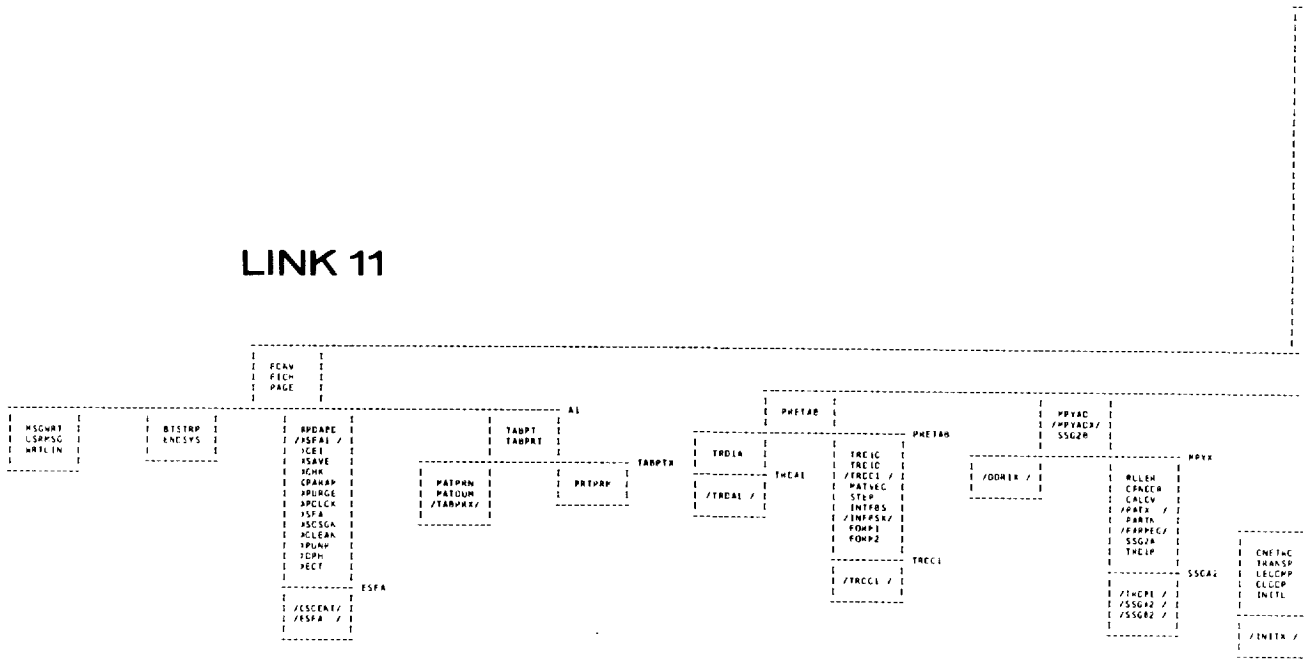
Figure 15. Overlay structure for links 7 and 9 on the IBM 7094/7040(44) DCS.

Figure 16. Overlay structure for links 10 and 11 on the IBM 7094/7040(44) DCS.

LINK 10



LINK 11



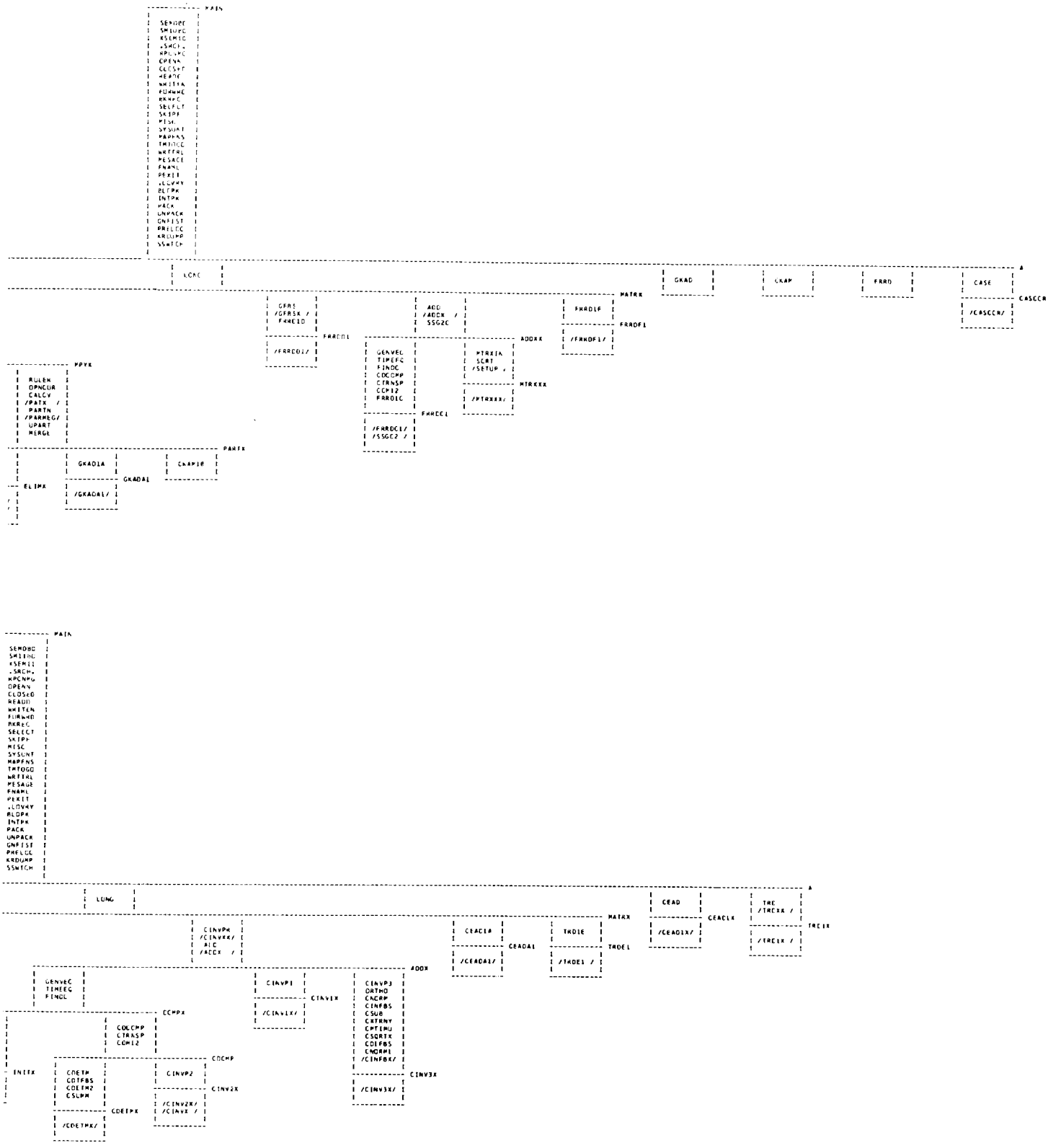
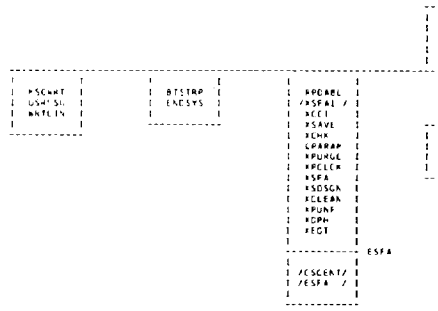


Figure 16. Overlay structure for links 10 and 11 on the IBM 7094/7040(44) DCS.

LINK 12



LINK 13

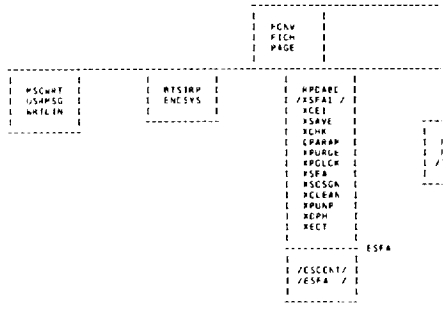
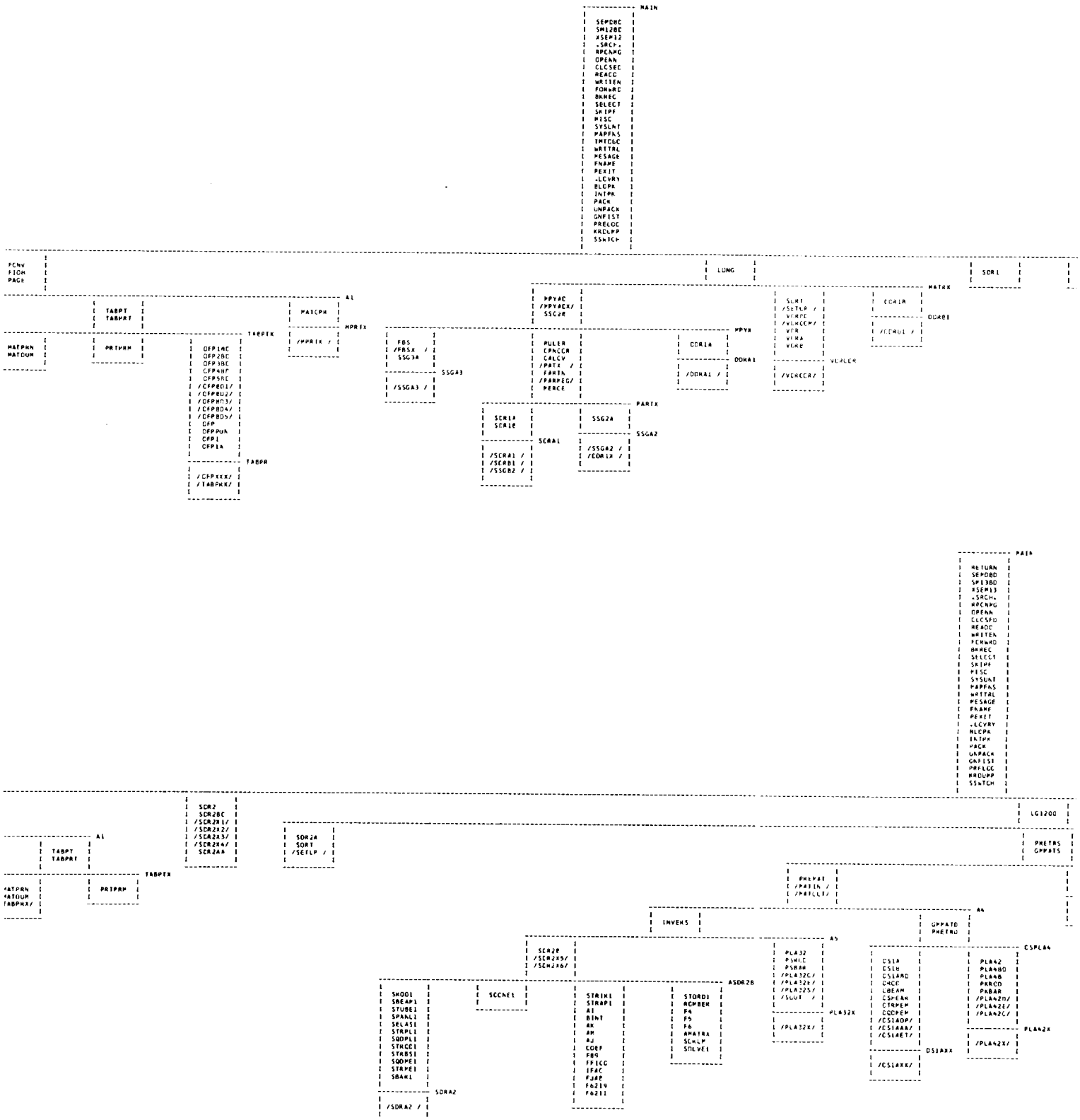


FIG. 17

ure for links 12 and 13 on the IBM 7094/7040(44) DCS.


MAIN
SEMDBC
SW128C
XSEM12
.SRCH.
MPCAMG
OPENN
CLCSEC
READC
WRITEN
FCNWRC
BKREC
SELECT
SKIPF
MISC
SYSUNT
MAPFAS
IMTCGC
WRTTRL
MESAGE
FNAME
PEXIT
.LCVRY
BLCPK
INTPK
PACK
UNPACK
GNFIST
PRELOC
KRDUPP
SSWTCH
FCNV
FIOH
PAGE
AL
TABPT
TABPRT
MATCPR
/MPRTX /
MPRTX
MATPRN
MATOUM
PRTPRM
TABPTR
OFP1NC
OFP2BC
OFP3BC
OFP4BC
OFP5BC
/OFPB01/
/OFPB02/
/OFPM03/
/OFPB04/
/OFPB05/
OFP
OFPPUA
OFP1
OFP1A
TABPR
/OFPXXX/
/TABPRX/
FBS
/FBSX /
SSG3A
SSGA3
/SSGA3 /
LONG
MPYAC
/MPYACX/
SSG2B
MPYR
RULER
CPNCCR
CALCV
/PATX /
FAMTA
/PARREG/
PERCE
CDR1A
ODRA1
/ODRA1 /
MATRX
SCRT
/SETLP /
VERPE
/VERECP/
VER
VERA
VERE
VERLER
/VERECR/
CDR1A
ODRB1
/CERROL /
SDR1
PARTX
SCR1A
SCR1B
SCRAL
/SCRA1 /
/SCRB1 /
/SSGB2 /
SSG2A
SSGA2
/SSGA2 /
/CDR1A /
PAIN
RETURN
SEMDBD
SW138D
XSEM13
.SRCH.
MPCNPG
OPENN
CLCSFD
READC
WRITEN
FCNWRD
BKREC
SELECT
SKIPF
MISC
SYSUNT
MAPFAS
WRTTRL
MESAGE
FNAME
PEXIT
.LCVRY
NLCPA
INTPK
PACK
UNPACK
GNFIST
PRFLCC
KRDUPP
SSWTCH
SCR2
SCR2BC
/SCR2X1/
/SCR2X2/
/SCR2X3/
/SCR2X4/
SCR2AA
SCR2A
SORT
/SETLP /
LG3200
PHETRS
GPPATS
AL
TABPT
TABPRT
TABPTR
MATPRN
MATOUM
TABPRX/
PRTPRM
PHEPAT
/PATIN /
/PATLCT/
AA
INVERS
GPPATD
PHETRD
A5
SCR2P
/SCR2X5/
/SCR2X6/
PLA32
PLACE
PSRAM
/PLA32C/
/PLA32X/
/PLA32S/
/SCOUT /
PLA32X
/PLA32X/
CSPLA4
CSIA
CSIB
CSIARD
CACE
LBEAM
CSPEAM
CTRMEM
CCOPEM
/CS1AOP/
/CS1AAA/
/CS1AET/
DS1AXX
/CS1AXX/
PLA42
PLA48D
PLA4B
PRRCD
PKBAR
/PLA42D/
/PLA42E/
/PLA42C/
PLA42X
/PLA42X/
ASDR2B
SHOD1
SREAP1
STUBE1
SPANL1
SELAS1
STRPL1
SQDPL1
STHCC1
STABS1
SQDME1
STRME1
SBAKL
SDRA2
/SDRA2 /
SCCNE1
STRIK1
STRAP1
AI
BINT
AK
AM
AJ
COEF
FB9
FFICC
IFAC
FJ88
F6219
F6211
STORD1
ACMBER
FA
F5
F6
AMATRX
SCHLP
STLVE1

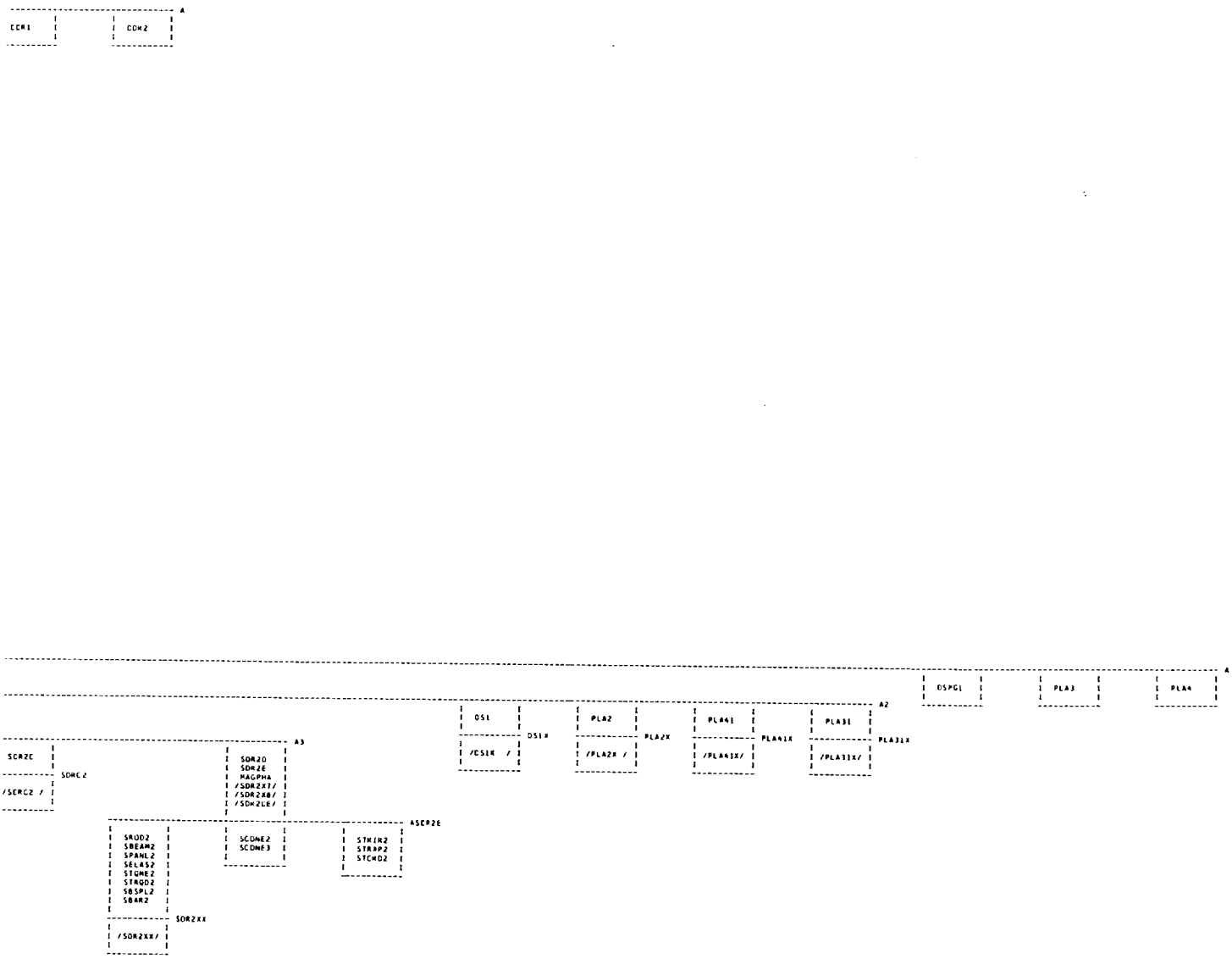


Figure 17. Overlay structure for links 12 and 13 on the IBM 7094/7040(44) DCS.

LINK 14

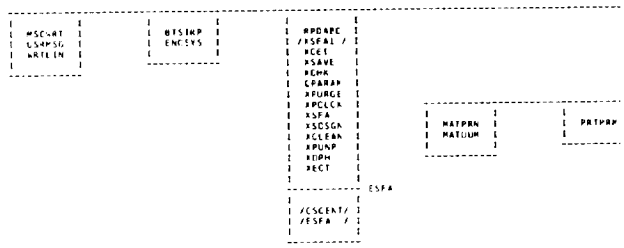


FIG. 18.
for link 14 on the IBM 7094/7040 DCS.

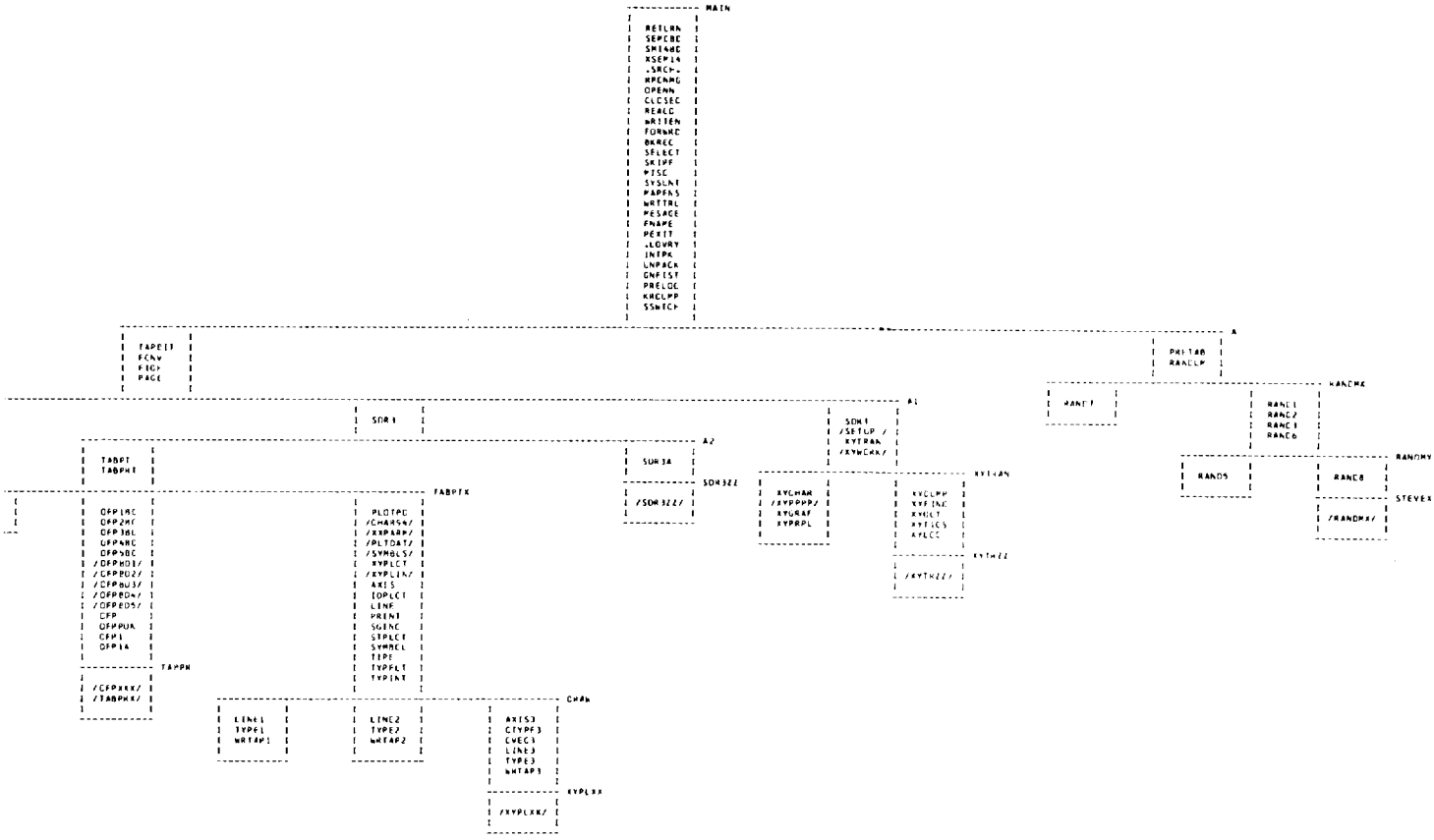


Figure 18. Overlay structure for link 14 on the IBM 7094/7040 DCS.

REFERENCE

1. Bluett, David E., SUBSYS, A Package for Improving the Efficiency of IBSYS. SHARE Distribution Agency, SDA#3369.

Figure 18. Overlay s

5.3 NASTRAN ON THE IBM SYSTEM/360 OPERATING SYSTEM (ØS)

5.3.1 Introduction

NASTRAN operates as a standard, single step job on the IBM System/360 class of computers under Operating System (ØS). NASTRAN is created as a Partitioned Data Set (PDS) with each data set member containing the structure and content of a NASTRAN link. One additional member, unique to the 360 NASTRAN configuration and residing in memory at all times, controls the loading of all other links and the necessary inter-link communication. This additional member, referred to as the Super-link, contains the FØRTRAN I/Ø routines and the ØS macros necessary to load and execute the subordinate functional links (NASTRAN links).

Within each functional link, a 360 dependent control routine contains the entry point for the Super-link call, the corresponding call to the proper XSEMi subroutine, plus the transfer points for FØRTRAN I/Ø, SEARCH and EXIT. This control routine is identical within each link with the exception of its entry point name and its XSEMi call which incorporates the link number (e.g., for link 6, the control routine entry point will be LINKNSO6, and the XSEMi call will be CALL XSEM6). The ØS PDS member name for the Super-link is "NASTRAN"; the member names for the functional links are the same as the control routine entry points (i.e., LINKNSO1, LINKNSO2, etc.).

To operate the NASTRAN system, the PDS containing the link members is attached to ØS by a JØBLIB control statement. Execution is initiated through an // EXEC PGM=NASTRAN job control statement (If the NASTRAN procedure is in PRØCLIB, only // EXEC NASTRAN is required. See section 5.3.5). This job control statement causes the member named NASTRAN (the Super-link) to be loaded and entered. The Super-link always calls LINKNSO1 (the Preface link) first. From this point on, the link execution sequence is determined by SEARCH calls from the operating functional link which returns control to the Super-link asking for another functional link. This sequence continues until a call to EXIT from within an operating link terminates NASTRAN and returns control to ØS. The downward calls from the Super-link to the function link members are accomplished via the ØS LINK macro-instruction. All intra-link calls are accomplished via standard FØRTRAN call statements within the XSEMi subroutine.

5.3.2 Input/Output

All NASTRAN I/Ø is performed through normal FØRTRAN I/Ø statements. Hard copy/printed output is generated by formatted output statements to FØRTRAN logical unit 6 (DDNAME=FT06F001). This output may be routed as desired although it normally appears as a system output (SYSØUT) class. Data card input is read by formatted input statements, a card (record) at a time, from FØRTRAN logical unit 5 (DDNAME=FT05F001). Again, the input source may be designated as desired although it normally appears as the system input (SYSIN) unit.

All functional module I/Ø is performed by calls to the various GINØ routines which in turn perform I/Ø via normal FØRTRAN I/Ø statements. GINØ blocks the data and generates standard FØRTRAN unformatted blocked reads, writes, backspaces, etc. The number of files available to GINØ is a function of the number of data definition (DD) cards within the NASTRAN job containing acceptable FØRTRAN DDNAMEs (FTxxF001). These DDNAMEs should designate only the first file (F001) even though GINØ may provide internal multi-file capability.

Since GINØ accesses the FØRTRAN units through the NASTRAN Executive tables XFIAT and FIAT, the GNFIAT (Generate FIAT) subroutine must interpret the available DD cards and place appropriate entries in these tables. GNFIAT extracts the task I/Ø table (TIØT) address from ØS through the EXTRACT macro. The TIØT contains pointers to the DD cards for the NASTRAN job. The DD cards are scanned for acceptable FØRTRAN DDNAMEs. Any DDNAME containing a unit number between 8 and 75 inclusive will be placed in the XFIAT/FIAT table (if FTxxF001 is the DDNAME, xx is the unit number). As each unit number is stored, a check of the device type code determines whether the XFIAT/FIAT magnetic tape flag should be set.

The SGINØ (Special GINØ) plotter output routines within the S/360 NASTRAN system function independently of other I/Ø. Unique DD cards within the Execution Deck Setup (see section 5.3.5) describe the required output tape file formats. Two separate plotter files may be generated, one on FØRTRAN unit 13 and the other on unit 14.

5.3.3 Link Switching

NASTRAN link switching on the 360 is performed by the Super-link and directed by the SEARCH entry within the control routine, LINKNSi, i = 01, 02, ..., 13. When the XSEMi subroutine within an operating functional link determines that the next module to be operated

resides in a link other than its own, the required link is requested through a call to SEARCH. The SEARCH call carries an argument naming the link requested. SEARCH branches back to the Super-link with the new link name. The Super-link executes the LINK macro to load the proper member and transfer control into this new NASTRAN link. Only one functional link at a time will occupy the 360 memory below the Super-link. Since the functional links are individual members of the NASTRAN Partitioned Data Set, the number of links is essentially onen ended.

5.3.4 Overlay Considerations and Open Core Implementation

Each NASTRAN functional link is created by the ØS Linkage Editor during the generation (SUBSYS) procedure (see section 5.3.6). Each link contains a zero level or root segment and the series of overlay segments necessary to NASTRAN operating logic. The special named common blocks which define the beginning of various open core areas are placed at their required location by linkage editor INSERT directives. An ØS overlay tree functions by automatically loading all segments in the branch between the calling segment and the called segment. Local FØRTRAN variables and common blocks residing within segments are not cleared at load time.

Several NASTRAN links contain a special type origin for the overlay tree. This origin is created by declaring a particular segment boundary to be at a Region Boundary. This region boundary automatically begins at the end of the longest branch in the previous region. Since many links contain a series of small functional module drivers of different lengths followed by a structure of matrix routines used by these drivers, a region boundary is usually utilized following the drivers. The following sketch illustrates a link structure with regions.

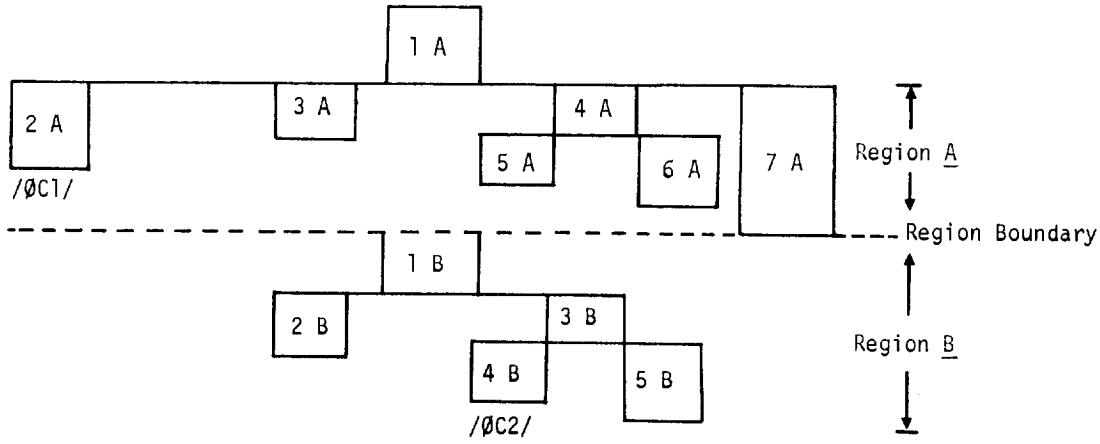


Figure 1. A tree diagram for a NASTRAN link on the IBM 360

Note that since segment 1B was requested to be begun at a region boundary, the ØS Linkage Editor placed its origin following the longest segment (7A) of the previous region.

Special caution must be exercised when operating within open core under a region structure. Operations within each region are independent of other regions. Thus, if a branch of segments within region B, say 1B and 3B, is in core because of previous calls, a call from region A to region B, say 5B, will not reload segments 1B and 3B. Therefore, if an open core area starting at ØC1/ were utilized, some of the programs within region B could be destroyed without the ØS loader's knowledge. Because of this possibility, all NASTRAN open core origins lie within a lower level region. Therefore open core starting areas such as /ØC1/ are not used in S/360 NASTRAN overlay structures.

Because NASTRAN must operate on most models of System/360 with a variety of memory sizes and ØS operating modes, a flexible method of utilizing all memory available to the job is incorporated. In both primary and multi-programmed environments, the ØS loader requests storage from the core memory available to the job through the ØS GETMAIN macro. ØS storage-management routines make sufficient core available, and the link is placed in memory. Core memory outside of that requested for the link is storage protected by the ØS system. An attempt to store into these protected areas causes an interrupt and job termination. The NASTRAN open core concept requires use of those areas that are available but protected (i.e., the area between the link and the region boundary). To remove this protected status, the NASTRAN initialization program within each functional link issues a conditional GETMAIN for all remaining memory within the job region. The return from this GETMAIN specifies the origin and the size of the block of core memory acquired. A small portion of this block is returned (via the FREEMAIN macro) to ØS for use as FØRTRAN I/Ø buffers and for other ØS functions. The remainder of this memory is made available to NASTRAN by adjusting the upper core address used by the NASTRAN CØRSZ function. All NASTRAN modules may thus utilize the maximum core memory provided to the job.

5.3.5 Execution Deck Setup

Running or executing the NASTRAN system on a System/360 computer once the generation procedure (see section 5.3.6) is complete requires some basic knowledge of the type of structural problem being solved and the type of output requested. In addition, the hardware configuration and capacities should be known in order to most adequately match the problem being solved to the computer.

The following procedure should provide the basic Job Control Language (JCL) necessary for all NASTRAN runs.

Table 1. Basic Job Control Language (JCL) for All NASTRAN Runs.

```
 1  //DEFAULT PROC  PRINT=A,SP1=1,SP2=5,BLK1=7204,BLK2=7208,VCT=1
 2  //NS EXEC PGM=NASTRAN
 4  //FT04F001 DD   SYSOUT=&PRINT
 5  //FT05F001 DD   DDNAME=SYSIN
 6  //FT06F001 DD   SYSOUT=&PRINT,SPACE=(CYL,(2,2))
 7  //FT07F001 DD   SYSOUT=B
 8  //FT08F001 DD   UNIT=(SYSDA,&VCT),SPACE=(CYL,(&SP1,&SP2)),VOL=(,,,&VCT),X
    //               DCB=(RECFM=V,LRECL=&BLK1,BLKSIZE=&BLK2,BUFNO=1,DEN=3)
 9  //FT09F001 DD   DDNAME=OPTP
10  //FT10F001 DD   DDNAME=NPTP
11  //FT11F001 DD   UNIT=SYSDA,SPACE=(TRK,1),LABEL=(,BLP),
    //               DCB=(RECFM=V,LRECL=&BLK1,BLKSIZE=&BLK2,BUFNO=1,DEN=2)
12  //FT12F001 DD   UNIT=SYSDA,SPACE=(TRK,1),LABEL=(,BLP),
    //               DCB=(RECFM=V,LRECL=&BLK1,BLKSIZE=&BLK2,BUFNO=1,DEN=2)
13  //FT13F001 DD   DDNAME=PLT1
13A //FT13F002 DD   SPACE=(TRK,1),LABEL=(02,BLP),VOLUME=REF=*.FT13F001,      X
    //               DCB=(RECFM=U,BLKSIZE=2400,BUFNO=1,DEN=1,TRTCH=E)
      .
13B //FT13F015 DD   SPACE=(TRK,1),LABEL=(15,BLP),                            X
    //               VOLUME=REF=*.FT13F001,DCB=*.FT13F002
14  //FT14F001 DD   DDNAME=PLT2
15  //FT15F001 DD   UNIT=SYSDA,SPACE=(TRK,1),DCB=*.FT08F001,LABEL=(,BLP)
20  //FT20F001 DD   UNIT=(SYSDA,&VCT),SPACE=(CYL,(&SP1,&SP2)),VOL=(,,,&VCT),X
    //               DCB=(RECFM=V,LRECL=&BLK1,BLKSIZE=&BLK2,BUFNO=1)
    //FT21F001 DD   UNIT=(SYSDA,&VCT),SPACE=(CYL,(&SP1,&SP2)),VOL=(,,,&VCT),X
    //               DCB=*.FT20F001,SEP=(FT20F001)
    //FT22F001 DD   UNIT=(SYSDA,&VCT),SPACE=(CYL,(&SP1,&SP2)),VOL=(,,,&VCT),X
    //               DCB=*.FT20F001,SEP=(FT20F001,FT21F001)
    //FT23F001 DD   UNIT=(SYSDA,&VCT),SPACE=(CYL,(&SP1,&SP2)),VOL=(,,,&VCT),X
    //               DCB=*.FT20F001,SEP=(FT20F001,FT21F001,FT22F001)
        .
        .
50  //FT50F001 DD   UNIT=(SYSDA,&VCT),SPACE=(CYL,(&SP1,&SP2)),VOL=(,,,&VCT),X
    //               DCB=*.FT20F001,SEP=(FT48F001,FT49F001)
SP  //SNAPSHOT DD   SYSOUT=&PRINT
NT  //NPTP     DD   UNIT=SYSDA,SPACE=(CYL,(1,2)),DCB=*.FT08F001,LABEL=(,BLP)
OT  //OPTP     DD   UNIT=SYSDA,SPACE=(TRK,1),DCB=*.FT08F001,LABEL=(,BLP)
P1  //PLT1     DD   UNIT=SYSDA,SPACE=(TRK,1),LABEL=(01,BLP),DCB=*.FT13F002
P2  //PLT2     DD   UNIT=SYSDA,SPACE=(TRK,1),LABEL=(,BLP),                   X
    //               DCB=(RECFM=U,BLKSIZE=2400,BUFNO=1,DEN=1)
```

This deck is presented in a form so that it may be added to an installation's Procedure Library (PRØCLIB) to permit easy recall, modification, and reuse of this NASTRAN JCL. Although the remaining discussion and examples will assume the preceding deck to be a catalogued procedure, similar JCL may, of course, be used within the job stream with each problem submission.

Each card or group of cards within the NASTRAN procedure is discussed by item below. The item numbers match those along the left margin of the preceding deck listing.

ITEM

1. Provides default value for various symbolic parameters within the procedure. PRINT=A, defines the output printer class as A. SP1=1 defines the primary space allocation request as 1 cylinder. SP2=5 defines the secondary space allocation request as 5 cylinders. BLK1=7204 defines the logical record length as 7204 bytes. BLK2=7208 defines the block size as 7208 bytes.

2. Loads member NASTRAN from NSTNLMØD data set attached to ØS by JØBLIB.

4. Defines unit 4 for CØNMSG (section 3.3.12) output to SYSØUT.

5. Defines unit 5 as the standard system input unit.

6. Defines unit 6 as the standard system output unit to SYSØUT. Note the SPACE item enlarges the spooling space for system output.

7. Defines unit 7 as the standard system punch unit. NASTRAN restart decks are output to this unit.

8. Defines unit 8 as a standard GINØ file. This unit will be placed in Executive table XFIAT as the Data Pool (PØØL) File.

9. Defines unit 9 to contain the Old Problem Tape (ØPTP), if one is to exist for this run. Also points to DDNAME ØPTP DD card for remaining definitions (see item ØT).

10. Defines unit 10 to contain the New Problem Tape (NPTP), if one is to exist for this run. Also points to DDNAME NPTP DD card for remaining definitions (see item NT).

11. Designates unit 11 for the User Master File (UMF). If this function is desired, this DD card must be overridden.

12. Designates unit 12 for the New User Master File (NUMF). If this function is desired, this DD card must be overridden.

13. Defines unit 13 to contain PLT1 (EAI 3500) plotter output. Also points to DDNAME PLT1 DD card for remaining definitions.

13A through 13B. Defines files 2 through 15 (note dot (.) for missing but similar cards) of unit 13 to contain additional plots on the PLT1 plot tape. Some plotters (e.g., the EAI 3500) require file marks between plots; ØS data management routines require a new DD card for each

file even though they reside on the same unit. The DCB parameter sets density at 556 BPI and even parity as required by the plotter hardware. If more than 15 plots are to be generated on PLT1, additional DD cards must be provided.

14. Defines unit 14 to contain PLT2 (SC4020) plotter output. Also points to DDNAME PLT2 DD card for remaining definitions.

15. Designates unit 15 as an auxiliary input unit. If this function is desired, this DD card must be overridden.

Units 8 through 15 reside in Executive table XFIAT and with the exception of unit 8 (PØØL) perform no NASTRAN function unless their UNIT parameters are altered to a physical tape by a DD override.

20 through 50. Defines units 20 through 50 as general GINØ files. These units are placed in the FIAT Executive table and are thus available for assignment by the Executive Segment File Allocator (XSFA) module. The data sets generated by these DD cards are requested to be separated from each other and to extend over multiple volumes if possible. The space size requests are controlled by the default values in the PRØC card or some execution time override.

SP. Defines the system output unit for any error dumps and backtraces produced by a NASTRAN run. (CALL PDUMP).

NT. Defines the attributes of the New Problem Tape (NPTP) attached to unit 10 (see item 10), and provides a convenient place for a UNIT parameter override when the Problem Tape is used.

ØT. Defines the attributes of the Old Problem Tape (ØPTP) attached to unit 9 (see item 9), and provides a convenient place for a UNIT parameter override when the Problem Tape is used.

P1. Defines the attributes of the PLT1 plot tape (for the EAI 3500 plotter) attached to unit 13 (see item 13) and provides a convenient place for a UNIT parameter override when this plotter output is requested. The DCB parameter sets density for 556 BPI and even parity required by the plotter hardware.

P2. Defines the attributes of the PLT2 plot tape (for the SC 4020 plotter) attached to unit 14 (see item 14) and provides a convenient place for a UNIT parameter override when this plotter output is requested. The DCB sets density for 556 BPI required by the plotter

hardware.

Assuming that the above NASTRAN procedure is in the installation's PRØCLIB, various examples of NASTRAN runs follow:

RUN A. Cold start, no UMF, no check pointing, no plotting, large memory available, 2314 disk packs.

```
//NASA     JOB (5048321,013,14-01),A.USER,MSGLEVEL=1,REGION=1000K
//JOBLIB   DD  DSNAME=NSTNLMOD,DISP=SHR
//P1       EXEC  NASTRAN
//NS.SYSIN DD *
      .
      .
/*
```

RUN B. Restart, no checkpointing, SC 4020 plots, small memory available, 2311 disk packs.

```
//NASA     JOB (5048321,013,14-01),A.USER,MSGLEVEL=1,REGION=250K
//JOBLIB   DD  DSNAME=NSTNLMOD,DISP=SHR
//P2       EXEC  NASTRAN,SP2=1,BLK1=3604,BLK2=3608
//NS.OPTP   DD UNIT=2400,VOLUME=SER=T324
//NS.PLT2   DD UNIT=2400,VOLUME=SER=T821
//NS.SYSIN DD *
NASTRAN  BUFFSIZE=903
      .
      .
/*
```

RUN C. Cold start, using UMF, checkpointing, EAI 3500 plots, medium memory available, 2314 disk packs.

```
//NASA     JOB (5048321,013,14-01),A.USER,MSGLEVEL=1,REGION=500K
//JOBLIB   DD  DSNAME=NSTNLMOD,DISP=SHR
//P3       EXEC  NASTRAN
//NS.FT11F001 DD UNIT=2400,VOLUME=SER=T491
//NS.NPTP   DD UNIT=2400,VOLUME=SER=T320
//NS.PLT1   DD UNIT=2400,VOLUME=SER=T822
//NS.SYSIN DD *
      .
      .
/*
```

Note: Units designated for plot tapes are always 7 track, therefore the installations SYSGEN'ed mnemonic for a 7 track unit should be used following the UNIT = keyword (e.g., 2400-2, 2400-7, 2400-7T, etc.). Units designated for any remaining tapes (PROBLEM TAPES, UMF, etc.) may be 7 or 9 track as the user desires and thus the appropriate unit mnemonic should be used.

The above examples assume that the tapes used do not possess meaningful labels. The parameter (Tijk) within the VØLUME=SER=Tijk is used only to identify to the computer operator the physical reel of tape to be mounted. Should it be desired to use ØS standard tape labeling, the LABEL keyword must be overridden to SL and a DSNAME keyword added. The tape labels and DSNAMES used should be unique for each physical reel.

NASTRAN data decks (Executive Control Deck, Case Control Deck, Bulk Data Deck) must be added to the above set-ups following the //NS.SYSIN DD * card. Note that Run B contains a special data card, NASTRAN, (see section 6.3.1) to reduce the GINØ buffer size to a value consistent with the storage blocks on the 2311 disk pack. The ratio between BUFFSIZE, BLK1 and BLK2 is determined as follows: 1) a BUFFSIZE corresponding to the desired physical record size on the file storage device is chosen (normally a full track on disk). BUFFSIZE is stated in words (4 bytes/word); 2) BLK2 is then set to 4 bytes less than BLKSIZE and BLK1 is set to 4 bytes less than BLK2. BLK2 and BLK1 are stated in bytes.

5.3.6 Generation of the NASTRAN Executable System (SUBSYSING)

The following card listing is a sample subset of the Job Control Language (JCL) and Linkage Editor control statements necessary to produce a functional link of the NASTRAN Executable System on the System 360. A dot (.) on a line indicates missing cards similar to those preceding.

```
//LKED EXEC PGM=IEWL,PARM='MAP,LIST,OVLY,DC,LET,SIZE=(1000K,100K)'
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(10,10))
//SYSLIB    DD DSNAME=SYS1.FORTLIB,DISP=SHR
//LIB       DD DSNAME=P2LOAD,DISP=SHR
//SYSLMOD   DD DSNAME=NSTNLMOD,DISP=OLD
//SYSLIN    DD *
  INCLUDE LIB(LINKNS04)
  INCLUDE LIB(ROOTSEG)
     .
  CHANGE NTRAN(RETURN)
  INCLUDE LIB(ENDSYS)
  INCLUDE LIB(GP4)
     .
  INCLUDE LIB(LOOP)
  INCLUDE LIB(EJDUM2)
  OVERLAY A
  INSERT   MCE1
  INSERT   MCE2
  OVERLAY A
  INSERT   SMP1
```

```
    •
OVERLAY MATRX(REGION)
INSERT   GMMATD
INSERT   SMA3A
INSERT   INVERD
    •
OVERLAY OEND(REGION)
INSERT   EJDUM2
ENTRY    LINKNS04
NAME     LINKNS04(R)
/*
```

The above statements assume that all of the necessary NASTRAN subroutines have been previously compiled and that their object modules exist as members of the Partitioned Data Set P2LØAD. Also, the output Partitioned Data Set (NSTNLMØD) for the complete executable system must be allocated and available. Note that all data sets listed in the JCL are assumed to be catalogued. The Linkage Editor control statements (i.e., those following SYSLIN) first select the necessary object modules from the P2LØAD library through the INCLUDE cards. Any cross references may be altered at the time by a CHANGE card. Note that the ENDSYS subroutine reference to NTRAN (a Univac 1108 dependent program) is altered to a RETURN. Following all necessary INCLUDEs, the overlay is structured through ØVERLAY and INSERT cards. The ØVERLAY card defines the beginning of a new segment, while the INSERT card places object modules within that segment. ENTRY and NAME cards are required to generate the proper entry address and data set member name for the functional link. A set of cards similar to this sample is required to generate each link member of the NSTNLMØD data set. These cards are included within the NASTRAN delivery package. After the creation of all necessary NASTRAN links, the NSTNLMØD data set will contain the complete Executable System. This data set may then be attached to ØS as a JØBLIB and the NASTRAN system executed (see section 5.3.5).

5.3.7 Machine Dependent Routines

The following utility routines necessary to NASTRAN operation must, by their nature, be implemented in a machine dependent manner. They are written in S/360 Assembly Language.

5.3.7.1 Clocks

An elapsed real (wall clock) time is available within the 360 NASTRAN system through a call to routine WHEN. A call to WHEN executes the ØS TIME macro, which returns time in 26 μ sec. increments from midnight. The initial call saves an initial time which is then used to produce elapsed times. WHEN returns time in integer seconds from the initial call.

A processor or task time is available within the 360 NASTRAN through a call to routine NØW. The NØW routine initializes and uses the interval timer on S/360 configurations with the timer option. NØW sets and interrogates the interval timer through the STIMER and TTIMER ØS macros. The STIMER macro is executed with the TASK option which specifies that the timer is to operate only when the task is active. This has the effect of measuring only the central processor operating time for the particular job. NØW returns time in integer seconds from the initial call.

5.3.7.2 DUMP and PDUMP

A FØRTRAN CALL PDUMP causes a subordinate call to the ØS SNAP macro to be executed. This subordinate call (to PDUMPX) contains a parameter designating whether the SNAP macro is to produce only a trace table or a complete Problem Program memory dump. A CALL PDUMP will return control to NASTRAN at the statement following the call. PDUMP outputs via DDNAME SNAPSHOT.

5.3.7.3 Date

The current date is available for NASTRAN problem identification and titling through a call to routine TDATE. A call to TDATE executes the ØS TIME macro which returns the Julian date. This date is converted to month, day, and year by TDATE. TDATE returns the date in a three word integer array as month, day, year, e.g., 3,27,69.

5.3.7.4 On-Line Messages

Eight-character messages may be printed on the operator console from a NASTRAN problem through a call to routine TYPWRT. TYPWRT moves the eight characters from the calling sequence to the message section of an ØS WTØ macro and executes the macro.

5.3.8 Special Error Codes from NASTRAN on the System/360

One of the following two types of error codes will accompany any abnormal end (ABEND) output from NASTRAN operating under ØS on the IBM System/360:

1. SYSTEM=XXX

   where XXX is an ØS code described in the IBM System Reference Library (SRL), Form C28-6631, titled IBM System/360 Operating System Messages and Codes.

2. USER=YY

where YY is a NASTRAN code described as:

Code	Explanation
1	A link name has been requested by SEARCH (3.3.6) which is beyond the range of those available. See the assembly listing for the link names available.
2	Insufficient DD cards have been provided to the NASTRAN job to fill the XFIAT (2.4.1.1) and FIAT (2.4.1.2) tables with files to the level required by the MAX ØPEN parameter in /SYSTEM/ (2.4.1.8).
3	Insufficient core memory has been provided beyond the NASTRAN link to provide space for the FØRTRAN I/Ø buffers of all MAX ØPEN files.
4	A core memory discontinuity exists between the end of the NASTRAN link and the beginning of open core found by the GETMAIN macro (see 5.3.4).
12	A NASTRAN subroutine issued a "CALL DUMP". See the system trace-back for the subroutine name.

5.4 NASTRAN ON THE UNIVAC 1108 (EXEC 8)

5.4.1 Introduction

The purpose of this section is to describe the unique operating features of NASTRAN on the Univac 1108 computer under the Exec 8 operating system.

The choice of EXEC 8 over EXEC II as the 1108 Executive System was made due to the added flexibility obtainable in the control language. The size and complexity of the NASTRAN system necessitates the use of a system allowing open ended expansion.

Much of the final checkout and validation of NASTRAN was done under EXEC 8. The use of catalogued files for the source, object, and absolute program files simplified the task of modifying and updating the NASTRAN system.

As can be seen in the following discussions, good advantage of the capabilities of EXEC 8 has been taken.

5.4.2 Input/Output

The machine dependent I/Ø routines in NASTRAN can be classified into three categories: 1) obtaining the logical files, 2) performing the actual I/Ø (other than plot output), and 3) generating the plot tapes.

GNFIAT is the subroutine responsible for identifying the logical files available for use by NASTRAN. For the Univac 1108, it is written in FØRTRAN. Its functional description can be found in section 3. Under EXEC 8, GNFIAT operates in conjunction with the system routine NTAB$. NTAB$ was reassembled for NASTRAN to allow access to logical units 1-40. Correspondingly a table is generated in GNFIAT to indicate the status of each of these units. The status code of 1 implies the unit is unconditionally available for insertion into the FIAT Executive table. A code of 2 means the unit can be used if it is not set up as a tape. A status code of 3 means the unit is unavailable. GNFIAT searches this table and inserts the available logical units into the FIAT.

In addition to the expanded NTAB$, NTRAN$ was assembled with the number of packets increased to 30. This was necessary due to the number of files used in NASTRAN at any one time.

The actual I/Ø on the Univac 1108 is controlled through the blocked I/Ø package, consisting of GINØ and its associated routines. The physical I/Ø directives are restricted to GINØIØ.

GINØIØ has branches on machine type (360, DCS or 1108) to transfer to the proper I/Ø instructions for a given machine. Under EXEC 8, NTRAN is used exclusively for I/Ø. The reader should consult the subroutine descriptions for GINØ and GINØIØ in section 3 for more information.

The final I/Ø dependent routine is SGINØ, whose responsibility is to generate the unformatted BCD and binary tapes for the plotters. The restriction in SGINØ is that the output records must be free of any NASTRAN or system control words. NTRAN satisfies these requirements and is used to perform the I/Ø.

5.4.3 Link Switching

A link on the Univac 1108 consists of a main program, MAINi, that calls subroutine XSEMi which in turn has calls to the subroutines corresponding to the modules residing in that link. Thus, a link is, in itself, a complete executable program.

The problem of link switching reduces to the problem of selectively controlling the execution of the various programs (links). This is accomplished under EXEC 8 by virtue of the TEST control card and by the executive request (ER) SETC$.

The option to either execute or skip a link is achieved by the following two system control cards:

```
@TEST          TNE/i/T3
@XQT           NASTRAN.LINKi
```

The right third of the condition word which is tested by the TEST command is set by an ER to SET$. By setting this third of the word to the value i, LINKi will be executed. If the value of the word is not i, the XQT instruction will be skipped.

A packet of these control cards consisting of TEST and XQT cards for each link gives the user the flexibility he desires. However, due to the capability of looping in NASTRAN, many of these packets are usually put on a catalogued file such that one ADD card can redirect the control card stream to the proper file.

The above description gives the external controls required for link switching. Internally, NASTRAN uses subroutines ENDSYS and SEARCH to direct the link switch. Their functional descriptions can be found in section 3. ENDSYS is used mainly for saving information required across links. SEARCH, however, is the routine which controls the link switch. Under EXEC 8,

this consists simply of setting the condition word to the value of the desired link, and terminating the execution of the current link.

Upon initiation of a new link, BGNSYS is called to restore previous information. An additional function of BGNSYS on the Univac 1108, controlled by a branch on machine type, is to restore drum pointers for all files off the load point. Since GINØ maintains a table of file positions, this table can be used to update the system drum pointers.

5.4.4 Overlay Considerations and Implementation of Open Core

The complex nature of the NASTRAN overlay picture and the peculiarities of the EXEC 8 loader pose some special problems on the 1108. The loader under EXEC 8 is a "block" or "segment" loader. A segment is loaded only when a subroutine within that segment is called. Also, when a segment is loaded, local data and common blocks are set to zero.

The implications of these features of the loader can be seen in the following overlay example, given in Table 1 and Figure 1. If common block /XX/ is initialized by the Block Data subprogram E, and subroutine A calls D directly, then D cannot reference the data in /XX/ since that segment has not been loaded. Also, if subroutine A stores data in /YY/, and subsequently calls subroutine C, /YY/ will be reset to zero as it is loaded.

Table 1. Example of Input to the MAP Processor for a NASTRAN Overlay on the Univac 1108.

```
@MAP,1                    SAMPLE

          SEG     AA
          IN      A
          SEG     BB*
          IN      B,XX
          SEG     CC*
          IN      C,YY
          SEG     DD*,CC
          IN      D
          SEG     CORE*,BB
          IN      DEFCØR,DFCØR
```

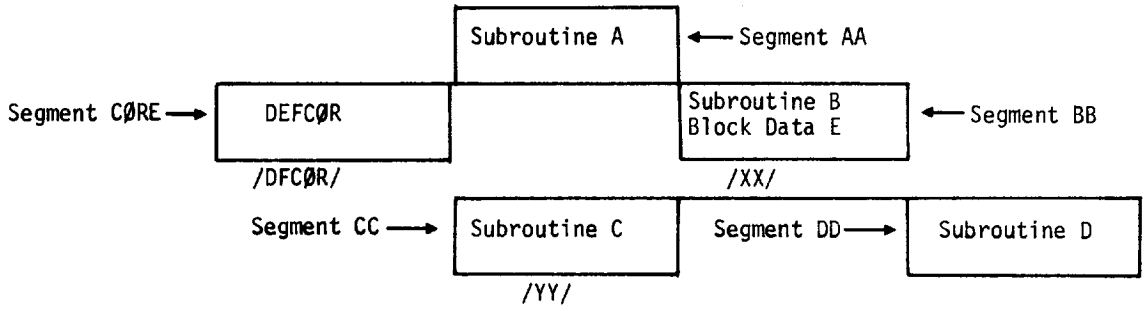


Figure 1. Example of NASTRAN overlay on the Univac 1108

As seen by the above example, some common blocks will have to be repositioned higher (in lower order core) in the overlay on the Univac 1108 than on the IBM 7094 and IBM 360 to protect NASTRAN from the loader.

The implementation of open core on the Univac 1108 consists of defining a segment of sufficient length to reserve core. This is seen in Figure 1 as segment CØRE. The FØRTRAN array in common block /DFCØR/ is dimensioned as large as possible. The last location of this common block is defined as the end of open core.

5.4.5 Execution Deck Setup

The following table gives a typical deck setup for NASTRAN on the 1108.

Table 2. Execution Deck Setup for NASTRAN on the Univac 1108

```
@RUN                                                                    001
@ASG,T   3,F///200                                                      002
@ASG,T   4,F///200                                                      003
@ASG,T   7,F///200                  . PØØL                              004
@ASG,T   8,F///200                  . ØPTP                              005
@ASG,T   9,F///200                  . NPTP                              006
@ASG,T   10,F///200                 . UMF                               007
@ASG,T   11,F///200                 . NUMF                              008
@ASG,T   12,F///200                 . PLT1                              009
@ASG,T   13,F///200                 . PLT2                              010
@ASG,T   14,F///200                                                     011
  .        .                                                             .
  .        .                                                             .
  .        .                                                             .
@ASG,T   35,F///200                                                     032
@HDG,N                                                                  033
@XQT     NASTRAN.LINK1                                                  034
NASTRAN DATA DECKS (Executive Control Deck, Case Control Deck,
                    Bulk Data Deck)
@TEST    TNE/1/T3                                                       035
@XQT     NASTRAN.LINK1                                                  036
@TEST    TNE/2/T3                                                       037
@XQT     NASTRAN.LINK2                                                  038
  .        .                                                             .
  .        .                                                             .
  .        .                                                             .
@TEST    TNE/14/T3                                                      047
@XQT     NASTRAN.LINK12                                                 048
@TEST    TNE/17/S6                                                      049
@JUMP    END                                                            050
@END:PMD,EB                                                             051
```

The following comments explain particular cards or groups of cards:

1. Card 001 consists of the standard RUN card required by the installation.

2. Cards 002-032 assign the logical files to Fastran or some other auxiliary storage device. These assign cards allow a maximum of 200 tracks or approximately 350,000 words for each file. This assignment can vary per run or per device used.

3. Cards 004-010 have a dual usage and can be used as user tapes. If a checkpoint is required then card 006 must be replaced with

```
@ASG,T          9,T,TAPE#
```

4. Depending upon the devices available, the assign cards can reference Fastran, drum, disk or tape.

5. Card 033 turns off the standard system heading.

6. Card 034 initiates execution of LINK1.

7. The NASTRAN data decks are inserted between cards 034 and 035.

8. Cards 035 to 050 constitute a packet of NASTRAN control cards. Many of these packets may be required for one run.

9. For user convenience, cards 002-034 and a series of packets each consisting of cards 035-050 can be catalogued. The execution deck setup can then consist of:

```
@RUN
@ADD            ASGCRDS.
NASTRAN DATA DECKS
@ADD            CØNTRØL.
```

where ASGCRDS and CØNTRØL are catalogued files.

5.4.6 Generation of the NASTRAN Executable System (Subsysing)

The initial generation of the NASTRAN executables or the modifying of a given link is easily accomplished. Since each link is considered as a seperate program, each link can be processed independently of any other link. Assuming the source and object decks are catalogued, a link of NASTRAN can be created by

```
@MAP,S        SØU.LINKi,NASTRAN.LINKi.
```

The symbolic MAP deck for LINKi resides as an element of the catalogued source file, SØU. The absolute element is stored in the catalogued file NASTRAN. A LIB card in the symbolic MAP directs the processor to the relocatable elements.

5.4.7 Machine Dependent Routines

The routines discussed in this section consist of those programs unique to the Univac 1108 or those which are implemented differently from other machines.

1. MAINi

MAINi is the main program for LINKi. Its sole function is to call DEFCØR and XSEMi.

2. DEFCØR

DEFCØR is responsible for defining a segment long enough to reserve core for NASTRAN. The last location of this segment is saved as the end of open core.

3. MAPFNS

In addition to the standard functions described in section 3, the following functions were added:

a) SETC(I) - Sets the condition word to I.

b) XCØRSZ(J) - Saves the address of J as the end of open core.

c) FACIL(UNIT,ID) - Sets ID = 1 if UNIT is assigned as a tape.

4. PPDUMP

PPDUMP provides an entry point for DUMP and PDUMP. When called, a DUMP of the open core is given and an error exit is forced if a DIAG 1 card is present in the Executive Control Deck.

5. UTIL

UTIL provides entry points for the NASTRAN clocks. These clocks return the elapsed CPU time in seconds.

5.5 NASTRAN ON THE CDC 6400/6600 (SCOPE 3)

5.5.1 Introduction

NASTRAN operates as a single job step on the CDC 6400/6600 computers under the SCOPE 3 Operating System. NASTRAN is created by the Linkage Editor (see section 5.6), a special program developed for the CDC 6400/6600 computers to provide compatibility with the other computing systems for which NASTRAN has been developed.

The overlay structure for NASTRAN on the CDC 6400/6600 is somewhat different from the other computing systems in that one link, Link 0, remains in core at all times during program execution. This link contains subroutines and Executive tables common to all links (e.g., GINØ, RDTRL, FNAME, /XXFIAT/, /XFIAT/, /SYSTEM/). Consequently, the primary functions of ENDSYS and BGNSYS (see section 3.3.5), which save and restore Executive tables, are not required. Link switching is accomplished in a manner similar to the IBM 360 through a CALL LINK. LINK is an entry point in the segment loader, which was developed in conjunction with the linkage editor.

The NASTRAN executable program normally exists as a sequential file on tape, although it may exist as an indexed file on disk. The first few records of the file contain an initial load program produced by the linkage editor. The remainder of the file is comprised of the NASTRAN links (one directory record per link plus one logical record per segment). To initiate the NASTRAN program, a control statement such as NASTRAN. is executed. This causes the CDC loader to load the initial program load portion of the NASTRAN file and transfer control to XBØØT (see section 5.6.1.1). The XBØØT program reads the remainder of the file, writes it on the disk in indexed form, reads Link 0 into central memory and transfers control to Link 0. Link 0 receives control through a small main program called NASTRAN which in turn calls Link 1 (the Preface link). Execution continues until the problem terminates through a call to PEXIT which in turn calls EXIT.

5.5.2 Input/Output

NASTRAN input/output is performed in three ways: (1) card input, printed output, and punched output is accomplished through standard FØRTRAN formatted input and output statements; (2) all other input/output except plots is accomplished through the GINØ routines which in turn call IØ6600 (see section 5.5.6.4) to perform reads and writes; and (3) plot output is accomplished through SGINØ (see section 3.4.51).

The number of files available to GINØ is a function of the MAXFILES parameter in the SYSTEM common block (see section 2.4.1.8). All NASTRAN files which have not been assigned to tape are treated as indexed files on the disk. A master index for each file is permanently maintained in central memory near the end of the field length (see Figure 2). The master index contains a disk pointer for each subindex record. The subindex records contain disk pointers for each of the blocked NASTRAN records. IØ66ØØ maintains the position of each file in the common block GINØ66. Subindex records are read or written at open and close time and during reading or writing when a subindex boundary is passed. Actual input/output is accomplished through XIØRTNS (see section 5.5.6.8), which is a CØMPASS subprogram. XIØRTNS communicates with the SCØPE Operating System through calls to CPC.

SGINØ on the CDC 6400/6600 functions independently of other I/Ø. Its point of commonality with other I/Ø routines is XIØRTNS, which is used to perform the physical writing of data.

Figure 1 depicts a complete buffer as used by GINØ, IØ66ØØ and XIØRTNS.

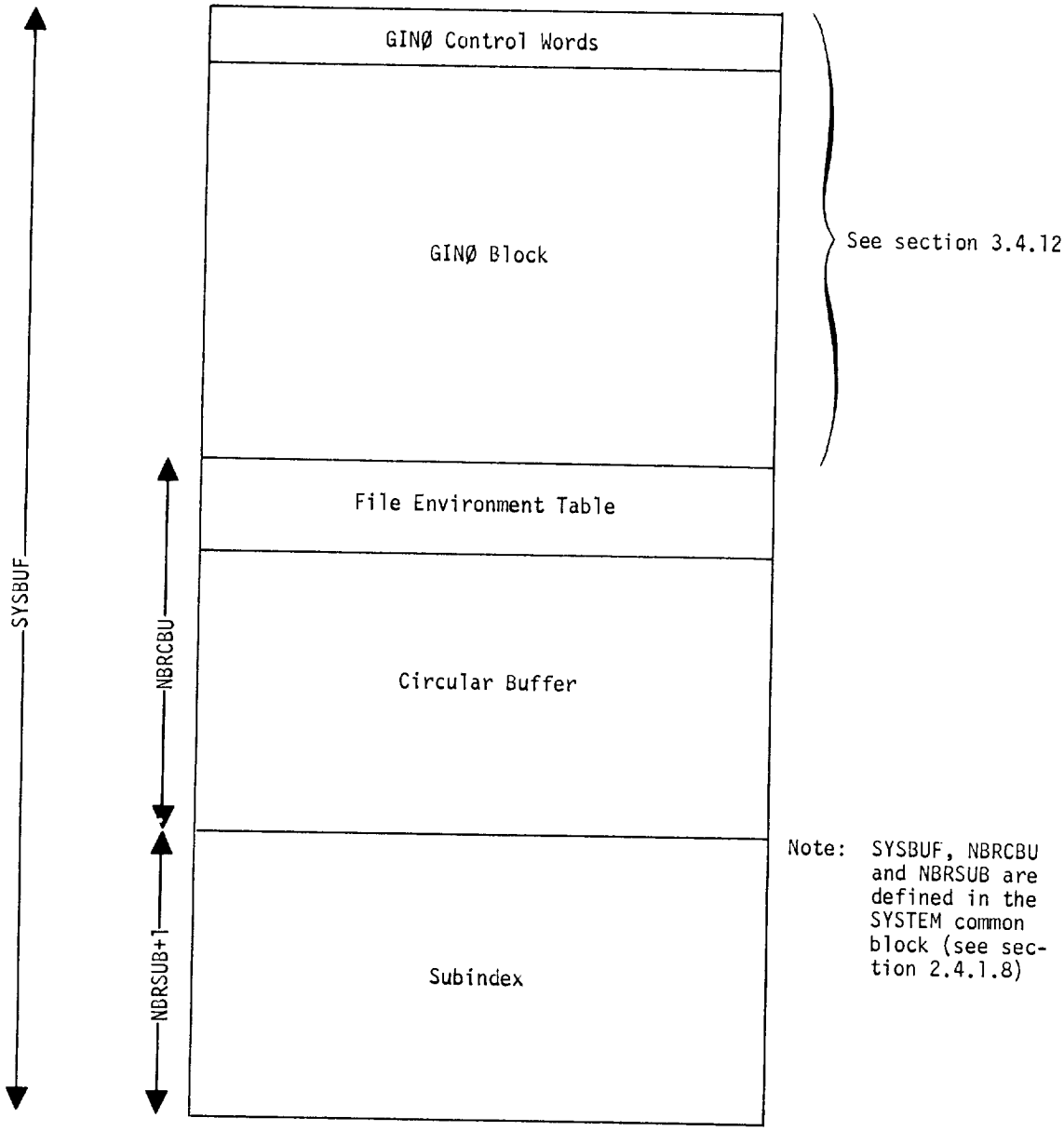


Figure 1. GINØ buffer on the CDC 6400/6600.

5.5.3 Layout of Core Storage

Figure 2 illustrates the layout of core storage on the CDC 6400/6600.

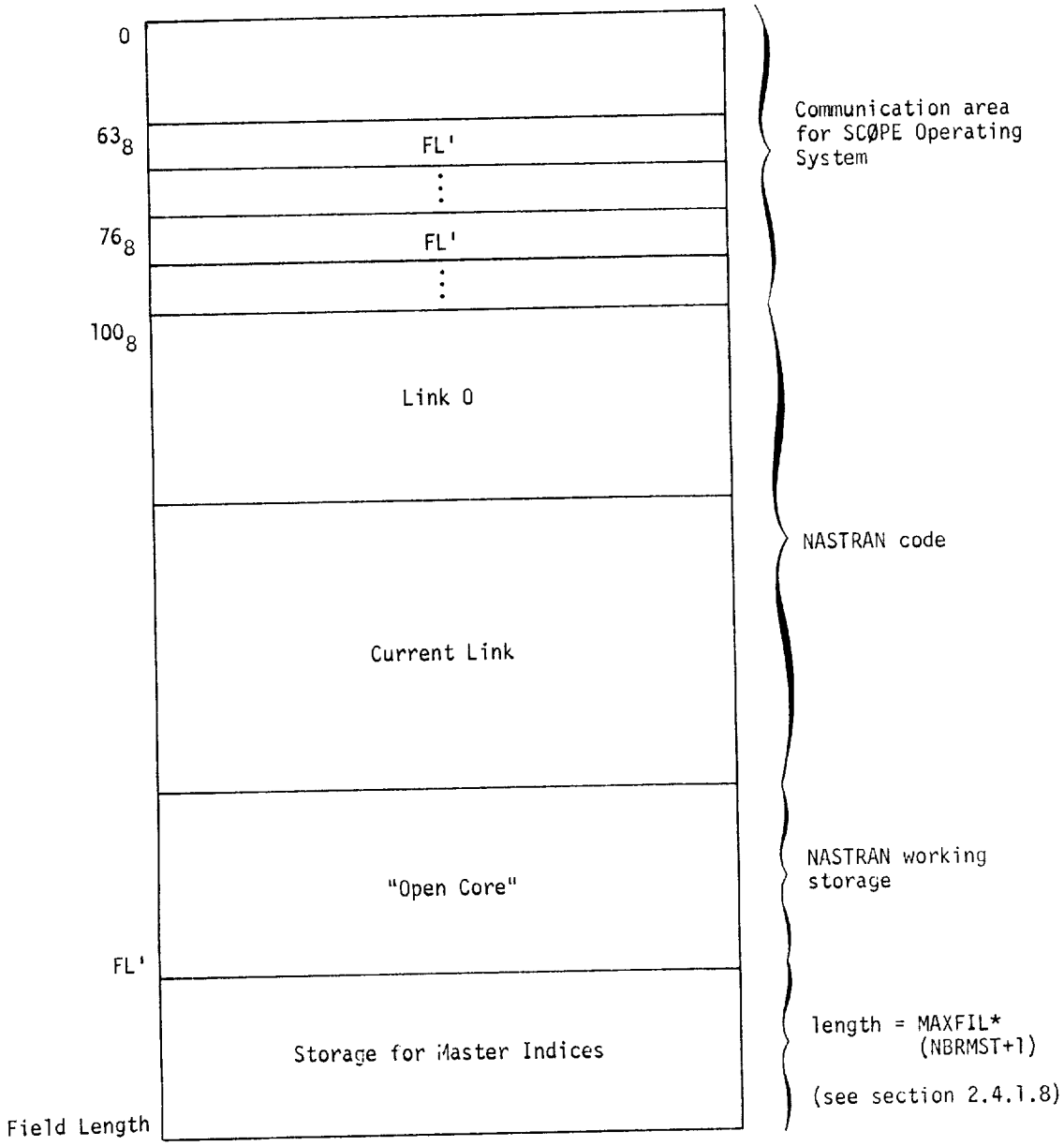


Figure 2. Layout of core storage on the CDC 6400/6600.

The primary function of GNFIAT (see section 3.3.4), which operates in Link 1, is to initialize the XFIAT and FIAT Executive tables. As a secondary function, GNFIAT reserves an area at the end of core storage for storage of the master index pointers for each file. The length of this reserved area is subtracted from the field length for the problem, and the available memory, FL', is stored in words 63_8 and 76_8. The length of open core is returned to a calling program by the CØRSZ function. CØRSZ subtracts the address passed to it from the contents of word 76_8. All NASTRAN modules may then utilize maximum core storage provided to the job.

The discussion in section 5.3.4 regarding overlay considerations on the IBM 360 applies equally to the CDC 6400/6600.

5.5.4 Execution Deck Setup

The following examples illustrate the control statements necessary to execute the NASTRAN program once the generation procedure (see section 5.5.5) is complete. The numbers in parentheses to the right of some of the control statements refer to notes, which follow the examples.

Problem Conditions					
Example	NASTRAN	Bulk Data Input Medium	Checkpoint	Restart	DD80 Plot
A	tape	cards	no	no	no
B	tape	UMF	no	no	no
C	disk(common)	cards	yes	no	yes
D	disk(common)	UMF	yes	no	no
E	tape	ØPTP	no	yes	no
F	tape	ØPTP	yes	yes	yes

Example A

(1) JØB,p,t,fl.

NØRFL.

NØMAP.

REQUEST NASTRAN,HI. reel#,RØL

NASTRAN.

DRØPFIL(NASTRAN)

$^{7}_{8}_{9}$ {NASTRAN Data Decks}

$^{7}_{8}_{9}$

$^{6}_{7}_{8}_{9}$

Example B

JØB,p,t,fl.

NØRFL.

```
      NØMAP.
      REQUEST NSTRN,HI. reel#,RØL
      REQUEST UMF,HI. reel#,RØL
(2)   NSTRN.CREATE(NASTRAN)
      DRØPFIL(NSTRN)
      DRØPFIL(UMF)
      7 8 9
        {NASTRAN Data Decks}
      7 8 9
      6 7 8 9
```

Example C

```
      JØB,p,t,fl.
      NØRFL.
      NØMAP.
      REQUEST NPTP,HI. SAVTP,RIL
      REQUEST PLT2,HI. SAVTP,RIL
(3)   ATTACH(NASTRAN)
      NASTRAN.ATTACH
      DRØPFIL(NPTP)
      DRØPFIL(PLT2)
      7 8 9
        {NASTRAN Data Decks}
      7 8 9
      6 7 8 9
```

Example D

```
      JØB,p,t,fl.
      NØRFL.
      NØMAP.
      REQUEST UMF,HI. reel#,RØL
```

```
REQUEST NPTP,HI. SAVTP,RIL
ATTACH(NASTRAN)
NASTRAN.ATTACH
DRØPFIL(UMF)
DRØPFIL(NPTP)
7
8
9
{NASTRAN Data Decks}
7
8
9
6
7
8
9
```

Example E

```
JØB,p,t,fl.
NØRFL.
NØMAP.
REQUEST ØPTP,HI. reel#,RØL
REQUEST NASTRAN,HI. reel#,RØL
NASTRAN.
DRØPFIL(ØPTP)
DRØPFIL(NASTRAN)
7
8
9
{NASTRAN Data Decks (including RESTART packet in the Executive Control Deck)}
7
8
9
6
7
8
9
```

Example F

```
JØB,p,t,fl.
NØRFL.
NØMAP.
REQUEST NASTRAN,HI. reel#,RØL
REQUEST NPTP,HI. SAVTP,RIL
REQUEST ØPTP,HI. reel#,RØL
```

```
REQUEST PLT2,HI. SAVTP,RIL
NASTRAN.
DRØPFIL(NASTRAN)
DRØPFIL(NPTP)
DRØPFIL(ØPTP)
DRØPFIL(PLT2)
7
8
9
{NASTRAN Data Decks (including RESTART packet in the Executive Control Deck)}
7
8
9
6
7
8
9
```

Notes

(1) On the JØB card: p = priority, t = time in octal seconds and fl = field length in octal words.

(2) In addition to executing the NASTRAN program, this control statement will cause the NASTRAN file to be declared a common file.

(3) This control statement attaches the common file to the job. If NASTRAN is executed as a common file, running time will be approximately 5-8 minutes faster in elapsed time and 20 seconds faster in CPU time.

5.5.5 Generation of the NASTRAN Executable System

Due to the differing formats of nonstandard returns and multiple entries in CDC FØRTRAN in comparison with Univac and IBM FØRTRAN's, the tape containing the symbolic card images of all the NASTRAN decks must first be converted to CDC compatible format by a utility program prepared for this purpose. The NASTRAN source tape is used as the input, and the CDC compatible source tape is written as output. The following control cards are used to execute this NASTRAN source conversion utility:

```
JØB,01,20,150000.
NØRFL.
REQUEST TAPE1,HI. reel#,RØL
REQUEST TAPE20,HI. SAVTP,RIL
```

```
CØPYBF(TAPE1,TAPE10)
REWIND(TAPE10)
INPUT.
REWIND(TAPE20)
7 8 9
{NASTRAN SOURCE CONVERSION UTILITY BINARY DECKS}
7 8 9
6 7 8 9
```

The output tape, SAVTP, is then compiled using the following control cards to produce a relocatable tape, NASTØBJ.

```
JØB,01,20,150000.
NØRFL.
REQUEST TAPE1,HI. reel#,RØL
REQUEST NASTØBJ,HI. SAVTP,RIL
CØPYBF(TAPE1,SØURCE)
REWIND(SØURCE)
RUN(S,,,SØURCE,,NASTØBJ,,1,1)
7 8 9
6 7 8 9
```

Having produced the relocatable tape, the NASTRAN executable tape can be generated using the NASTRAN CDC 6400/6600 Linkage Editor. This process with examples may be found in section 5.6.

5.5.6 Machine Dependent Routines

The following utility routines necessary to NASTRAN operation must, by their nature, be implemented in a machine dependent manner. Certain of the routines have been written in CØMPASS language, and the remainder are in FØRTRAN language.

5.5.6.1 MAPFNS (CØMPASS)

The MAPFNS deck embodies a set of 22 utility functions and routines. These are as follows:

Logical Functions

ANDF (logical product of two words)

ØRF (logical sum of two words)

XØRF (logical difference of two words)

CØMPLF (complement of a word)

LSHIFT (left shift of a word)

RSHIFT (right shift of a word)

Utility Functions

CØRSZ (returns length of "open core")

CØRWDS (returns difference of two addresses + 1)

LWØRDS (returns difference of two addresses. CDC 6600 only)

LØCF (returns the value of an address. CDC 6600 only)

Utility Routines (CDC 6600 only)

XSTØRE (stores an array at an absolute position in core)

XFETCH (fetches an array from an absolute position in core)

XJUMP (transfers control to an absolute location in core)

ZAP (stores zeros between specified locations in core)

RECØVRY (initializes PP routine RCV)

LINK2O. (provides a special call to Link 20, the NASTRAN exit link)

XCØMMØN (executes the CØMMØN macro and returns the status of a file)

FIELDLN (returns the field length assigned to the job)

FLUSH (makes a special call to SYSTEM. to empty output buffers)

TDATE (returns month, day, year, time)

KLØCK (returns current value of CPU clock)

DAYTIME (returns current time on wall clock)

5.5.6.2 CØNMSG (FØRTRAN)

The CØNMSG routine enters messages in the DAYFILE denoting the times of the initiation and completion of modules during NASTRAN execution.

5.5.6.3 GNFIAT (FØRTRAN)

GNFIAT initializes the XFIAT and FIAT tables and reserves storage for the master indices

(see section 5.5.2).

5.5.6.4 IØ6600 (FØRTRAN)

Calls from GINØ to GINØIØ are renamed to IØ6600 on the CDC 6400/6600. Section 5.5.2 contains a discussion of the function of IØ6600.

5.5.6.5 DUMP and PDUMP (FØRTRAN)

A call to PDUMP produces a trace back listing and, if a DIAG 1 card appeared in the Executive Control Deck, a memory dump is given. PDUMP returns to the calling program. DUMP operates similarly except terminates with a CALL EXIT.

5.5.6.6 PEXIT66 (FØRTRAN)

PEXIT66 is called from PEXIT only. It terminates activity on all NASTRAN files. Tapes are unloaded. Space assigned to disk files is evicted. PEXIT66 terminates with a CALL EXIT.

5.5.6.7 SGINØ (FØRTRAN)

SGINØ is used by the plot routines to write plot tapes. Actual writes are accomplished by SGINØ through calls to XIØRTNS.

5.5.6.8 XIØRTNS (CØMPASS)

The XIØRTNS deck embodies a set of utility routines which communicate with the SCØPE Operating System through CPC to accomplish various file operations.

Entry Points

XØPEN (constructs FET and initiates activity for file)

XCLØSE (terminates activity for file)

XEVICT (evicts space assigned to a disk file)

REINDX (resets index pointers in the FET)

XWRITE (writes a portion or a complete logical record)

XREAD (reads a portion or a complete logical record)

XREWIND (repositions file to load point)

XBKREC (repositions file one logical record backwards)

XBKPREC (repositions file one physical record backwards)

XFRDREC (repositions file one logical record forwards)

XREQST (issues REQUEST macro)

5.5.6.9 GN66BD (FØRTRAN)

GN66BD is a BLØCK DATA subprogram which initializes the GINØ66 common block with names of each of the possible NASTRAN files.

5.6 THE CDC 6400/6600 LINKAGE EDITOR

5.6.1 Introduction

5.6.1.1 Concept of the Linkage Editor

The linkage editor has been designed to provide an efficient and effective means of utilizing core storage for medium to large programs. The existing loader for the CDC 6400/6600 systems has the following disadvantages:

1. Only two levels of overlay are provided beyond the root segment.
2. An overlay segment must be explicitly called. Consequently, the overlay structure must be known when the program is coded.
3. An overlay segment may be entered at one point only. Consequently, downward calls are extremely limited.
4. No facility exists to explicitly position named common blocks.
5. Loading of overlay segments is accomplished from a sequential file, thus providing unnecessary search time.

The CDC 6400/6600 Linkage Editor in conjunction with its partner, the Segment Loader, overcomes these disadvantages in the following ways:

1. An unlimited number of overlay levels is provided.
2. The programmer describes the overlay structure to the linkage editor after the program is coded. The linkage editor provides implicit segment loading.
3. Complete communication between all levels of overlay is maintained.
4. Linkage editor control statements may be used to explicitly position subprograms and named common blocks.
5. The overlay segments are maintained in an indexed file. Consequently, every segment is immediately available to the segment loader.

As may be seen from Figure 1, the primary input sources to the linkage editor include:

1. Object decks (relocatable binary decks)
2. Control statements

3. A call library from which unsatisfied external references are resolved.

Another source of input (not shown in Figure 1) is a file containing executable links from a previous linkage editor run. This feature allows changes or additions of links while not altering previous links to which no changes are required.

The file produced by the linkage editor contains three portions:

1. A sequence of object decks suitable for loading by the CDC loader. The main program in this sequence, named XBØØT, reads the remainder of the file containing the executable links and writes it on the disk as an indexed file. XBØØT reads Link 0 into central memory and transfers control to the entry point which initiates execution of the problem program. This sequence of decks is terminated by a null record.

2. Three records:

   (1) Link 0 directory record;

   (2) Link 0 symbol dictionary containing entry points and common blocks in Link 0 and their associated addresses;

   (3) Link 0 executable record.

3. A directory record for each succeeding link and one logical record per segment containing executable instructions and data.

This sequence of records is terminated by a directory record which contains the word ENDLINKS.

Link 0 remains in central memory at all times during program execution. Link 0 contains no overlay segments. The linkage editor supplies a routine named XLØADER when Link 0 is constructed. XLØADER accomplishes the loading of segments and links when requested. Segment load requests are supplied automatically by the linkage editor through tables called ENTAB$ (see section 5.6.3.2) which are written as a part of the text for each segment which may require additional segment loading. An additional table, SEGTAB$ (see section 5.6.3.2), which is constructed by the linkage editor as a part of the root segment of every link, is used by XLØADER to facilitate segment loading.

Major divisions of a program are links. Each link consists of a self-contained overlay structure and might be thought of as a complete program in itself. All routines in a link communicate freely with Link 0 routines. Consequently, Link 0 may be thought of as logically

belonging to every link. For many programs, a single link in addition to Link 0 will be sufficient. Because of its size, however, NASTRAN has been divided into 14 links.

5.6.1.2 Functions of the Linkage Editor

The basic function of the linkage editor is the linking of separately assembled or compiled subprograms into a link. The link is in a format suitable for loading and execution.

Although this linking or combining of subprograms is its primary function, the linkage editor also:

1. Incorporates subprograms from a library file to resolve undefined external references.
2. Constructs an overlay program in a format suitable for loading and execution.
3. Rearranges control sections and renames external references as directed by linkage editor control statements.
4. Reserves storage for common control sections generated by CØMPASS and FØRTRAN.
5. Provides processing options and diagnostic messages.

5.6.1.3 Subprogram Linkage

Processing by the linkage editor makes it possible for the programmer to divide his program into several subprograms which may be separately assembled or compiled. The linkage editor combines these subprograms into a link with contiguous storage addresses. The link is written in an indexed file. The linkage editor can process more than one link in a single job step. Each link is written with a unique link number.

5.6.1.4 Input Sources

Input to the linkage editor consists of one or more sequential files (libraries) containing subprograms in relocatable format as produced by CØMPASS or FØRTRAN, and linkage editor control statements contained in INPUT, the standard input file.

External references that are undefined after processing all subprograms cause the automatic library call mechanism to search for subprograms that will resolve the references. When these subprograms are found, they are processed by the linkage editor and become part of the link.

5.6.1.5 Programs in an Overlay Structure

To minimize main storage requirements, the programmer can organize his program into an overlay structure by dividing it into segments according to the functional relationshp of the subprograms. Two or more segments that need not be in main storage at the same time can be assigned the same storage addresses, and can be loaded at different times. The programmer uses linkage editor control statements to specify the relationship of segments within the overlay structure.

5.6.1.6 Options and Diagnostic Messages

The linkage editor can produce a storage map and a cross-reference table that show the arrangement of control sections in the link and how they communicate with each other. A list of the linkage editor control statements that were processed can be produced. Additionally, processing options that negate the effect of minor errors and specify the disposition of input and output files can be specified by the programmer.

Throughout processing by the linkage editor, errors and possible error conditions are printed. Serious errors cause a link not be written on the output file.

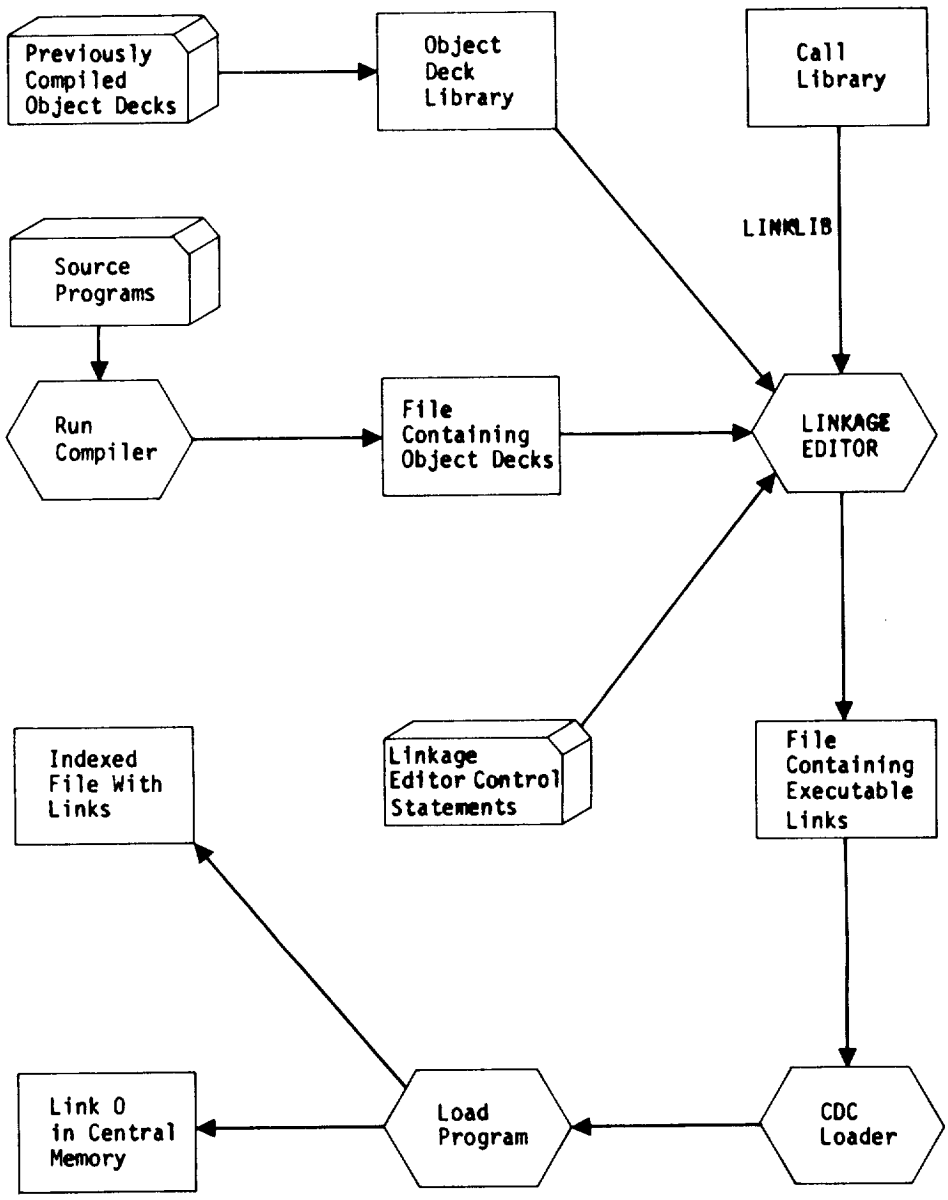


Figure 1. Linkage editor processing.

5.6.2 Preparing for Linkage Editor Processing

5.6.2.1 Object Decks

An object deck (relocatable binary deck), which is produced by the FØRTRAN compiler or CØMPASS assembler, consists of the following tables:

1. Program Identification and Length (PIDL), which defines a) subprogram identification and length and b) each of the common blocks referenced by the subprogram;
2. Entry Point (ENTR), which defines each of the entry points to the subprogram and its relative location;
3. Text (TEXT), which defines instructions and data for relocation;
4. Fill (FILL), which contains information to relocate previously defined address fields (in particular, references to common blocks);
5. Link (LINK), which indicates each external reference by the subprogram and its relative location;
6. Replication (REPL), which permits the repetition of a block of data.

Any other tables which may be contained in an object deck are ignored by the linkage editor. For a complete description of the format of these tables, see the SCØPE Reference Manual, Appendix D.

5.6.2.2 Libraries

All object decks that are to be processed by the linkage editor are contained in libraries. A library is a sequential file (which may reside on tape or disk) consisting of one or more logical records with one object deck per logical record.

A file named LINKLIB must always be defined for linkage editor processing. This file contains object decks for automatic library call plus object decks which are required in constructing the initial load portion of the output file.

There is no theoretical limit to the number of libraries which may be defined for linkage editor processing. Subprograms of the same name may appear in more than one library.

5.6.3 Designing an Overlay Program

5.6.3.1 Overlay Tree Structure

In order to place a program in an overlay structure, the programmer should be familiar with the following terms:

1. A control section consists of all instructions and data defined for a subprogram or a common block.

2. A segment is the smallest functional unit (one or more control sections) that can be loaded as one logical entity during program execution.

3. A path consists of a segment and all segments in the same region between it and the root segment (first segment). The root segment is a part of every path in every region. When a segment is in main storage, all segments in its path are also in main storage.

4. A region is a continguous area of main storage within which segments can be loaded independently of paths in other regions. An overlay program can be designed in single or multiple regions.

5. A link is a collection of one or more segments which comprise a logical subdivision of the program. Link 0 (consisting of one segment only) is in main storage at all times. It is the first link to receive control when execution of the program is initiated. The root segment of any other link resides in main storage at all times that that link is being executed. An overlay program must consist of at least one link other than Link 0.

6. A tree is the graphic representation that shows how segments can use main storage at different times. It does not imply the order of execution.

The design of an overlay program requires the organization of the control sections of the program in an overlay tree structure. The tree structure is developed considering:

1. The amount of available main storage.

2. The frequency of use of each control section.

3. The dependencies between control sections.

4. The manner in which control should pass within a path, from one path to another, and from one region to another.

When the programmer has determined the overlay structure for a program, he prepares ØVERLAY, INSERT and REGIØN statements that will segment the program in that manner. The use of these control statements is described in section 5.6.4.

5.6.3.2 Overlay Characteristics

During execution of an overlay program, the segment loader uses tables that were generated by the linkage editor and incorporated into the text of applicable segments. Since these tables are an integral part of the program, their size must be considered when planning the use of available main storage. These tables are described as follows.

1. Input/Output Control Table

There is one Input/Output Control Table (LINKO$) in the root segment of Link 0 only which contains a File Environment Table (FET), a circular buffer, a master index and a subindex. The LINKO$ table is used by the segment loader to read requested segments into central memory. LINKO$ is the first control section in Link 0. Its size is determined as follows:

$$\text{Length in words} = \text{PARAM}(1) + \text{PARAM}(4) + \text{PARAM}(5) + 4 \ .$$

Section 5.6.4.2 contains definitions of the parameters.

2. Segment Table

There is one Segment Table (SEGTAB$) in the root segment of each link except Link 0. The segment table is used to keep track of: (1) the relationship of the segments in the program; (2) which segments are in main storage or scheduled to be loaded; (3) the main storage address and length of each segment; and (4) the entry address of the link.

SEGTAB$ is the first control section in the root segment of each link. Its size is determined as follows:

$$\text{Length in words} = n + 2,$$

where n is the number of segments in the link.

3. Entry Table

There can be an Entry Table (ENTAB$) in each segment of the program. The loader

uses the entry table to determine the segment to be loaded when an external reference is made to a segment not in the path.

An entry table may be produced as the last control section of a segment. An ENTAB$ entry is created for a symbol to which control is to be passed. The symbol is defined in a segment not in the path. The size of ENTAB$ is determined as follows:

$$\text{Length in words} = 3n + \sum_{i=1}^{n} \delta_i,$$

where n is the number of unique external references not in the path and $\delta_i = MAX(m_i-6,0)$, m_i = number of arguments for each external reference not in the path.

4. Dump Control Word

In the text produced by the linkage editor for each segment, a uniquely formatted word which identifies the control section is written immediately prior to each control section. This word is recognized by the storage dump routine XDUMP in order to produce relative addresses for each control section.

5.6.3.3 Overlay Communication

There are two ways in which the programmer can have his program request the overlay facilities of the segment loader:

1. By a CALL statement (FØRTRAN language) or RJ instruction (CØMPASS language) which causes a segment to be loaded and control to be passed to the symbol defined in that segment.

2. By a CALL LINK(N) (FØRTRAN language) or the equivalent in the CØMPASS language, where N is the link number, which causes segment one (the root segment) of the requested link to be loaded and control to be passed to the symbol named on the linkage editor control statement ENTRY.

5.6.3.4 Reserving Storage

In FØRTRAN and CØMPASS the programmer can create control sections that reserve main storage areas containing no data or instructions. Referred to as "common", these control sections are produced by the language translator. These common areas are either named or blank (unnamed).

During processing, the linkage editor collects these common areas. If more than one blank common area is found, the largest blank common area is contained in the link. If two or more

common areas have the same name, the largest common area having that name is reserved in the link. All references to a common area (named or blank) refer to the largest area defined. This largest area is the one which is retained.

If the linkage editor encounters data or text for the same common area in more than one subprogram, only data from the first subprogram encountered are retained and a diagnostic message is generated for any subsequent data definitions.

When object decks which reference common areas are to be placed in an overlay structure, the linkage editor automatically "promotes" the common areas to the root segment (unless otherwise directed by an INSERT control statement, see section 5.6.4.8). The position of a promoted common area in relation to other control sections in the root segment is generally unpredictable.

Note: Blank common is treated by the linkage editor as a named common block with the special name BLANK.. and is listed on the storage map with this name. Consequently, it is possible to position this control section with the statement INSERT BLANK...

5.6.3.5 Processing Options

1. List of control statements

The linkage editor automatically produces a listing of all control statements unless the programmer selects the NØLIST option in the LINKEDIT statement (see section 5.6.4.2). In the latter case, only the LINKEDIT, LIBRARY and ENDLINKS statements are listed (see sections 5.6.4.2, 5.6.4.3 and 5.6.4.12 respectively for details).

2. Storage map and cross-reference table

The linkage editor automatically produces a storage map of each link unless the programmer selects the NØMAP option in the LINKEDIT statement. For each segment, the storage map lists the control sections in ascending order according to their assigned address. Included with each control section is a list of all entry point names and assigned addresses.

When the XREF option in the LINKEDIT statement is specified, the linkage editor produces a table of all references to each entry point in the link. Additional options (PARAM(7) parameter, see section 5.6.4.2) allow the table to be extended to include all references from the link to LINK 0 entry points and an additional table of all external references from each subprogram to be produced.

The NØMAP and XREF options are mutually exclusive. Therefore, if XREF is selected, NØMAP is ignored and a storage map is produced.

3. The LET option

When the LET option of the LINKEDIT statement is selected, the linkage editor disregards all errors except two and writes the link on the output file. The two errors which preclude the link from being written are: (1) an undefined entry point to the link; and (2) insufficient storage space to form the link to be written.

5.6.4 Linkage Editor Control Statements

5.6.4.1 General Statement Format

All linkage editor control statements are coded from the following possible forms:

operation	operand
VERB	a, b(c), KEYWØRD, KEYWØRD = a, KEYWØRD = b(c),
	KEYWØRD(i) = n, a = a, b(c) = a,n

where

a is an unsubscripted symbol,

b is a subscripted symbol,

c is a subscript symbol,

KEYWØRD is an explicit name or option,

i is an integer subscript,

n is an integer value.

The operation field must contain the name of the operation to be performed. The operand field must contain one or more symbols or subscripted symbols (except REGIØN, END and ENDLINKS which have no operands). Operands in the operand field are separated by a comma or blank (or both). Two or more symbols within parentheses are similarly separated. A keyword must be written exactly as shown.

The operation field begins with the first nonblank column on the card. The operand field is separated from the operation field by at least one blank column.

The LINKEDIT and LIBRARY control statements may be continued on subsequent cards by coding a comma as the last nonblank column. The continuation begins with the first nonblank column of the succeeding card. These two control statements are the only ones which may be continued.

5.6.4.2 The LINKEDIT Statement

The LINKEDIT statement specifies input and output file names and status, processing options and size characteristics of the link(s) to be link-edited.

operation	operands
LINKEDIT	INFILE = name(a),ØUTFILE = name(b), LET,NØLIST,NØMAP,XREF,PARAM(i) = n

INFILE — specifies the name of a file which was previously produced by the linkage editor. INFILE is named only when previously link-edited links are to be updated.

a — must be coded T or C. T specifies that the file is a sequential file on tape or disk. C specifies that the file is an indexed file on disk and, therefore, a common file as defined by SCØPE.

ØUTFILE — specifies the name of the file on which the initial load program and the links will be written. ØUTFILE must always be named.

b — must be coded T or C. The links are written internally by the linkage editor on an indexed file. If C is coded, the links are written directly on the file specified by ØUTFILE, which must be a common file. If T is coded, the initial load program and the links are copied from the internal file to the specified sequential file (tape or disk).

LET — directs the linkage editor to ignore the effect of most errors.

NØLIST — directs the linkage editor to suppress the listing of control statements.

NØMAP — directs the linkage editor to suppress storage maps.

XREF directs the linkage editor to list a table of external references to each entry point in each link.

Keyword	Description	Default Value
PARAM(1)	Length of FET + circular buffer for all files used by the linkage editor	530
PARAM(2)	Maximum number of object decks in all libraries	1000
PARAM(3)	Maximum size of any table in an object deck	500
PARAM(4)	Maximum number of links	32
PARAM(5)	Maximum number of segments in any link	128
PARAM(6)	Maximum length (in words) of any control section for which text is defined	5000
PARAM(7)	Additional options if XREF is selected =1: produce a table of references from each subprogram =2: list all references to entry points in Link 0 =3: provide both of the above	0

Placement: The LINKEDIT statement must be the first statement of the input record. Only one such statement may appear in a job step.

Notes: (1) The files named on INFILE and ØUTFILE may be the same; however, if so, their status must be the same (i.e., both T or both C)

(2) If XREF is selected, NØMAP is ignored

(3) If XREF is not selected, PARAM(7) is ignored

Examples:

```
LINKEDIT  ØUTFILE=LINKS(T),LET,XREF
LINKEDIT  INFILE=ØLDLNKS(T),ØUTFILE=NEWLNKS(T),NØLIST,PARAM(1)=1050
LINKEDIT  INFILE=ABS(C),ØUTFILE=ABS(C),PARAM(6)=8000
LINKEDIT  ØUTFILE=MYFILE(T)
```

5.6.4.3 The LIBRARY Statement

The LIBRARY statement names each file which may possibly occur on INCLUDE control statements (see secton 5.6.4.5).

operation	operands
LIBRARY	$name_1$,$name_2$,...

name — specifies the name of a sequential file containing object decks.

Placement: The LIBRARY statement must appear immediately after the LINKEDIT statement. Only one such statement may appear in a job step.

Example:

```
LIBRARY  MASTER,NEWDCKS
```

5.6.4.4 The LINK Statement

The LINK statement specifies the link number and directs the linkage editor to initiate processing of a link.

operation	operand
LINK	n

n — is a nonnegative integer specifying the link number.

Placement: The first LINK statement must appear immediately following the LIBRARY statement. Subsequent LINK statements must appear immediately following

an END statement. If Link 0 is being processed, it must be the first link to be processed.

Example:

```
LINK  0
```

5.6.4.5 The INCLUDE Statement

The INCLUDE statement directs the linkage editor to include the named object deck from the specified library in the link currently being processed.

operation	operands
INCLUDE	name(deck,BLKDATA(comname))

name	specifies the name of a sequential file which was previously defined in the LIBRARY statement.
deck	specifies the name of an object deck contained in the file.
BLKDATA	indicates that the deck to be included is a BLØCK DATA sub-program.
comname	is the name of the first mentioned common block in the BLØCK DATA sub-program.

Placement: An INCLUDE statement may appear in any position between the LINK and END statements. Subprograms will be included in the order in which INCLUDE statements are encountered.

Examples:

```
INCLUDE  MASTER(MAIN)
INCLUDE  NEWDCKS(MØD1,MØD2,MØD3)
INCLUDE  MASTER(BLKDATA(CØM1),BLKDATA(CØM2))
```

5.6.4.6 The REGIØN Statement

The REGIØN statement indicates the beginning of a new region.

operation
REGIØN

Placement: The REGIØN statement follows statements which define the overlay structure for a previous region, and it indicates the beginning of a new region.

5.6.4.7 The ØVERLAY Statement

The ØVERLAY statement indicates the beginning of an overlay segment.

operation	operand
ØVERLAY	name

name is the symbolic origin of a segment. The symbol is not related to external symbols in the link.

Placement: The ØVERLAY statement may appear in any position between the LINK and END statements. If a REGIØN statement is coded, an ØVERLAY statement must appear immediately following the REGIØN statement. An ØVERLAY statement may not be coded for Link 0.

Examples:

1. Single region structure (no REGIØN statement necessary)

```
INCLUDE  MASTER(SUB1)
INCLUDE  MASTER(BLKDATA(CØM1))
ØVERLAY  ALPHA
INCLUDE  NEWDCKS(MØD1)
ØVERLAY  BETA
INCLUDE  NEWDCKS(MØD2,MØD3)
ØVERLAY  BETA
```

```
INCLUDE  MASTER(PRØGA)
ØVERLAY  ALPHA
INCLUDE  MASTER(PRØGB,PRØGC)
```

Figure 2 depicts a tree diagram for the above example.

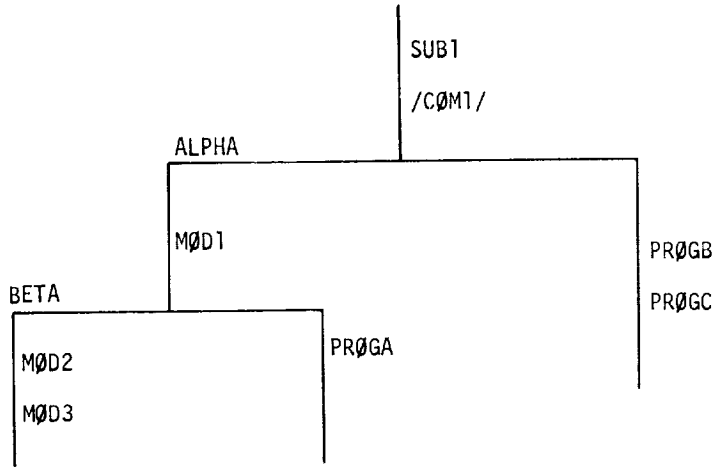


Figure 2. Tree diagram of a single region overlay structure.

2. Multiple region structure

```
INCLUDE  ØBJ(A,B,C)
ØVERLAY  ØNE
INCLUDE  DECKS(AA,BB)
ØVERLAY  TWØ
INCLUDE  ØBJ(D)
ØVERLAY  TWØ
INCLUDE  DECKS(CC,DD)
ØVERLAY  ØNE
INCLUDE  ØBJ(E,F,G)
REGIØN
ØVERLAY  THREE
INCLUDE  ØBJ(I,J)
```

```
ØVERLAY  THREE
INCLUDE  DECKS(EE)
ØVERLAY  FØUR
INCLUDE  DECKS(FF,GG)
ØVERLAY  FØUR
INCLUDE  ØBJ(K)
```

Figure 3 depicts a tree diagram for the above example.

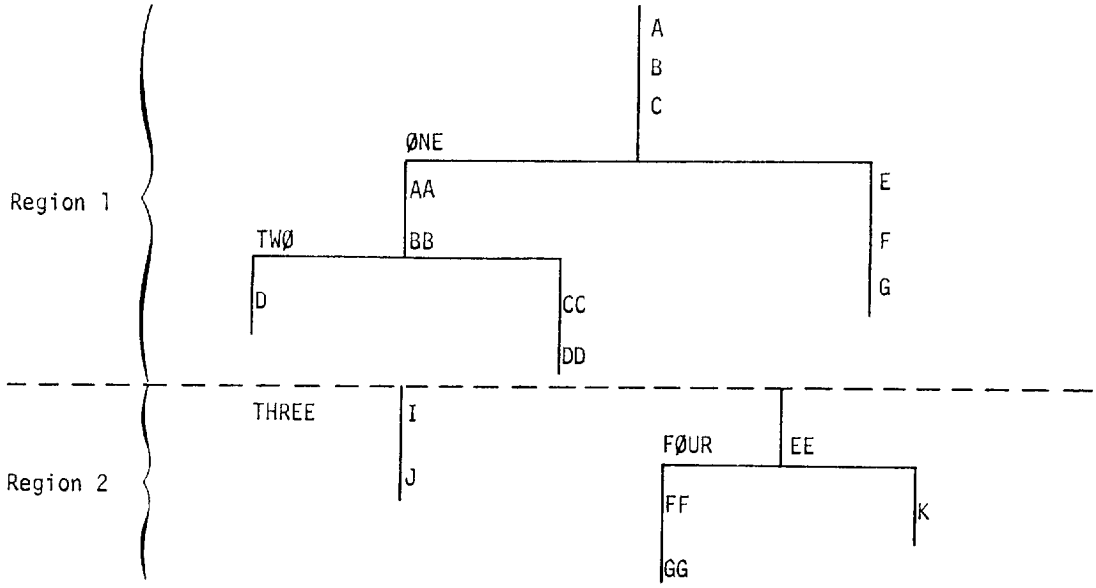


Figure 3. Tree diagram of a multiple region overlay structure.

5.6.4.8 The INSERT Statement

The INSERT statement positions control sections in overlay segments.

operation	operands
INSERT	name

name — specifies the name of a control section that is to be positioned.

Placement: The INSERT statement is placed in the control statement sequence following an ØVERLAY statement that defines the segment in which the control section is to be placed.

Caution: A control section should be named only once on an INSERT statement since a control section can appear only once within a link. If the same control section appears on more than one INSERT statement, the last statement encountered will control positioning and previous statements will be ignored.

Examples:

```
INSERT  CØM1
INSERT  CØM2,CØM3
```

5.6.4.9 The RENAME Statement

The RENAME statement changes external references to a symbol to a new symbol either globally (throughout the link) or locally (within a subprogram).

operation	operand	
RENAME	oldname=newname	[global]
RENAME	oldname(subprogram)=newname	[local]

oldname is the symbol to which an external reference exists.

newname is the symbol to which the external reference is to be made.

subprogram is the name of the subprogram in which the change is to be made.

Placement: The RENAME statement may appear in any position between the LINK and the END statements.

Notes: (1) Only one rename may be coded on a single statement.

(2) No error occurs if no references are made to oldname. If reference is made to oldname and newname is not specifically included, the automatic call logic will be invoked.

Examples:

```
RENAME  SQRT=SQRTXX
RENAME  DUMP(MØD1)=RETURN
```

5.6.4.10 The ENTRY Statement

The ENTRY statement specifies the symbolic name of the entry point in the link to which control will be transferred after thelink is loaded.

operation	operand
ENTRY	name

name — is the symbol defining the entry point for the link. The name must be defined in the root segment of the link. For Link 0, name must be the name of a main program. For any link other than Link 0, name must be the name of a subroutine.

Placement: The ENTRY statement may appear in any position between the LINK and the END statements.

Examples:

```
ENTRY  MAIN
ENTRY  SUB1
```

5.6.4.11 The END Statement

The END statement specifies the conclusion of control statements for the current link being processed.

operation
END

Placement: The END statement must appear following all statements which define the link. There must be one END statement for each LINK statement.

5.6.4.12 The ENDLINKS Statement

The ENDLINKS statement defines the end of all processing by the linkage editor.

operation
ENDLINKS

Placement: The ENDLINKS statement must immediately follow an END statement. One such statement is required.

5.6.5 Examples of Linkage Editor Processing

In the following examples, it is assumed that the file containing the call library (LINKLIB) and a file containing the linkage editor program (LINKEDT) are contained on separate magnetic tapes. These examples are intended to illustrate various deck setups.

Conditions of Problem			
Example	Source Program	Previously Compiled Decks	Execution
A	yes	in input stream	yes
B	no	on tape	no
C	no	input stream and on tape	yes

Example A

```
JØB,01,100,60000.
NØRFL.
NØMAP.
RUN(S,,,,,NEW)
REWIND(NEW)
CØPYBR(INPUT,ØLD,n)
REWIND(ØLD)
REQUEST LINKEDT,HI. reel#,RØL
REQUEST LINKLIB,HI. reel#,RØL
CREATE(LINKS)
LINKEDT.
DRØPFIL(LINKLIB)
DRØPFIL(LINKEDT)
LINKS.ATTACH
RELEASE(LINKS)
7
8
9
   {FØRTRAN or CØMPASS source programs}
```

```
7
 8
  9
    {n object decks}
LINKEDIT  ØUTFILE=LINKS(C)
LIBRARY  NEW,ØLD
LINK  0
(1) RENAME  SYSTEM=SYSTEM.
    {INCLUDE statements}
ENTRY  entry point
END
LINK  1
RENAME  SYSTEM=SYSTEM.
    {INCLUDE, ØVERLAY, etc. statements
ENTRY  entry point
END
ENDLINKS
7
 8
  9
    {data for problem program}
7
 8
  9
6
 7
  8
   9
```

Example B

```
JØB,01,100,60000.
NØRFL.
NØMAP.
REQUEST ØBJECT,HI. reel#,RØL
REQUEST LINKLIB,HI. reel#,RØL
REQUEST LINKEDT,HI. reel#,RØL
REQUEST LINKFIL,HI. SAVTP,RIL
LINKEDT.
DRØPFIL(ØBJECT)
```

```
DRØPFIL(LINKLIB)
DRØPFIL(LINKEDT)
DRØPFIL(LINKFIL)
7
8
9
LINKEDIT  ØUTFILE=LINKFIL(T),LET,XREF,PARAM(7)=2
LIBRARY  ØBJECT
LINK  0
RENAME  SYSTEM = SYSTEM.
    {INCLUDE statements for Link 0}
ENTRY  entry point
END
LINK  1
RENAME  SYSTEM=SYSTEM.
    {INCLUDE, ØVERLAY, etc. statements for Link 1}
ENTRY  entry point
END
ENDLINKS
7
8
9
6
7
8
9
```

Example C

```
JØB,01,200,70000.
NØRFL.
NØMAP.
CØPYBR(INPUT,ØBJ,n)
REWIND(ØBJ)
REQUEST MASTER,HI. reel#,RØL
REQUEST LINKLIB,HI. reel#,RØL
REQUEST LINKEDT,HI. reel#,RØL
```

```
REQUEST LINKFIL,HI. reel#,RIL
LINKEDT.
DRØPFIL(MASTER)
DRØPFIL(LINKLIB)
DRØPFIL(LINKEDT)
LINKFIL.
DRØPFIL(LINKFIL)
7
8
9
   {n object decks}
LINKEDIT  INFILE=LINKFIL(T),ØUTFILE=LINKFIL(T),PARAM(6)=90000
LIBRARY  MASTER,ØBJ
LINK  2
RENAME  SYSTEM=SYSTEM.
   {INCLUDE, ØVERLAY, etc. statements for Link 2}
ENTRY  entry point
END
ENDLINKS
7
8
9
   {data for problem program}
7
8
9
6
7
8
9
```

In Example A, the output of the linkage editor is written on a common file and executed from that file. This method is most efficient for "compile-and-go" type code check runs.

In Example B, the output of the linkage editor is written on tape. This would be the most common form of a run in which most of the coding errors have been eliminated and the executable program is saved on tape for subsequent repeated executions.

In Example C, it is assumed that a previously link-edited file exits (created, for example in Example B) and that it is desired to add a new link (or modify an existing link). In this case, the input and output files are both the same (although this is not necessary). The output

of the linkage editor is written on tape and the problem program is executed from this tape file.

Note: (1) To avoid possible conflict with a user (viz. NASTRAN) named common with the name SYSTEM, the CDC system routine has been reassembled with the name SYSTEM. and has been placed on LINKLIB with the name SYSTEM.. In order that other library routines may be properly linked, the statement RENAME SYSTEM=SYSTEM. is recommended for all links.

In the example illustrated in Figure 4, the following linkage editor control statements will produce the indicated overlay structure:

```
LINKEDIT  ØUTFILE=LINKS(T)
LIBRARY  LIBA,LIBB
LINK  0
RENAME  SYSTEM=SYSTEM.
INCLUDE  LIBA(MAIN)
INCLUDE  LIBB(UTIL1,UTIL2)
INCLUDE  LIBA(UTIL3)
ENTRY  MAIN
END
LINK  1
RENAME  SYSTEM=SYSTEM.
INCLUDE  LIBB(START,MØD1)
ØVERLAY  A
INCLUDE  LIBA(MØD2)
INCLUDE  LIBB(MØD3)
INSERT  CØM1
ØVERLAY  A
INCLUDE  LIBA(MØD4)
ØVERLAY  B
INCLUDE  LIBB(MØD5)
INCLUDE  LIBA(MØD6)
INSERT  CØM2
ØVERLAY  B
INCLUDE  LIBB(MØD7)
```

```
INSERT  CØM3
ENTRY   START
END
ENDLINKS
```

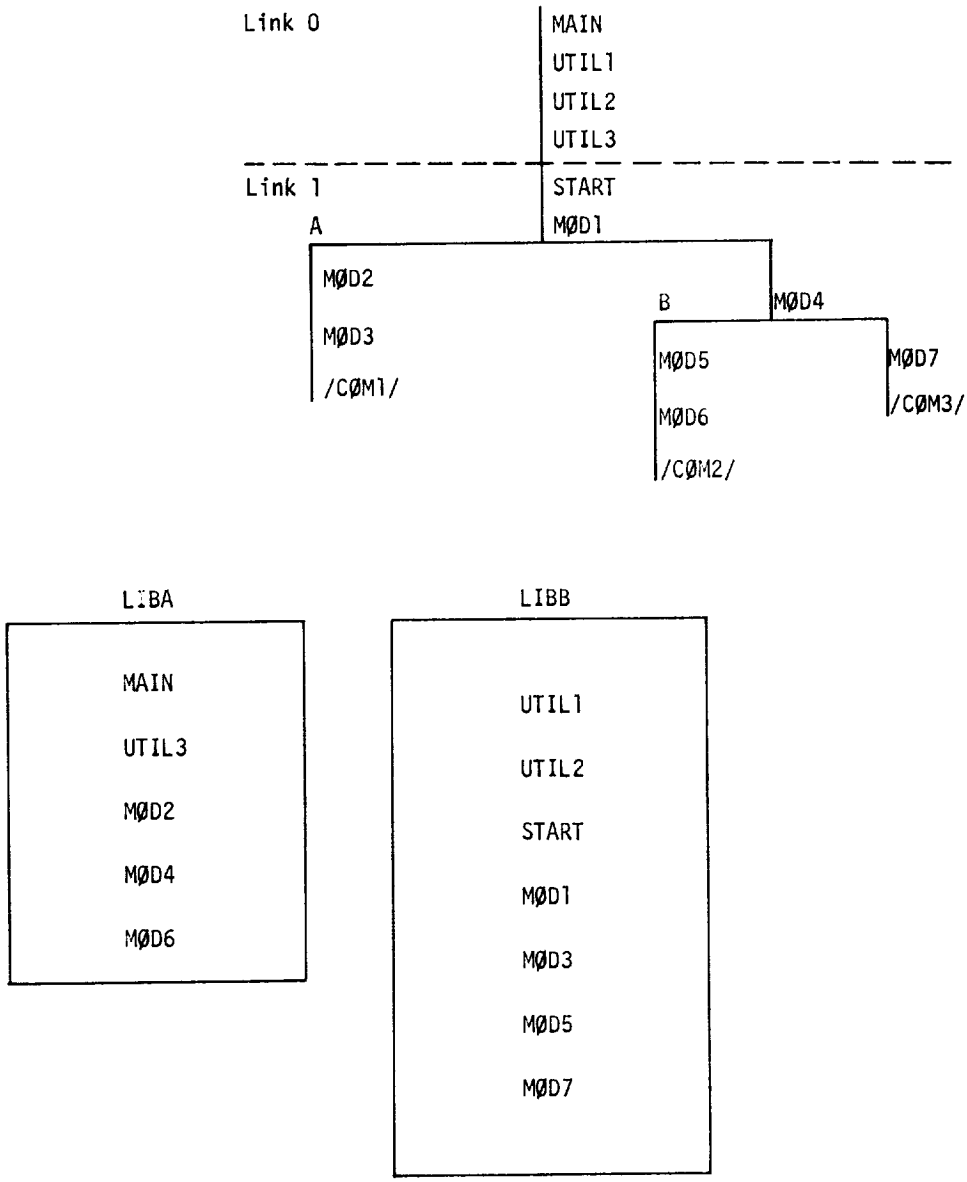


Figure 4. Example of a two-link overlay structure.

5.6.6 Storage Requirements for the Linkage Editor

Figure 5 illustrates the layout of core storage for the linkage editor. For the discussion below, it is assumed that the linkage editor has not itself been link-edited. A link-edited version of the linkage editor is available. A memory saving of approximately $4000_{10}(10000_8)$ words results.

The principal open-ended table is the Symbol Chain Table. A three-word entry is created in this table for each subprogram name, entry point, common block and unique external reference not in the path. For a link other than Link 0, a three-word entry for each entry point and common block in Link 0 is also created. A conservative estimate for the requirements of this table is as follows:

Link 0: length in words = 4* (no. of entry points + common blocks),

Link ≠ 0: length in words = 6* (no. of entry points + common blocks)
+3* (no. of entry points + common blocks in Link 0).

The largest table is likely to be the Working Storage Table. It must hold all instructions and data for the largest control section for which text is defined. If this figure is not known, a linkage editor run can be made. The storage map will be printed even if the link is not written. A scan of the lengths listed (in octal) will identify the largest control section. Note that common blocks for which no data are defined are not to be used in defining the maximum.

Field length for the linkage editor may be estimated from the following:

$$\text{field length}_{10} = 15000 + \text{MAX}(10*N,2000) + \text{MAX}(T,2000) + 3*\text{PARAM}(1) \quad ,$$

where

N = number of subprograms defined on INCLUDE statements,

T = length of largest subprogram or common block for which instructions or data are defined,

and

PARAM(1) is defined in section 5.6.4.2.

If default values for the linkage editor are used, a program of less than 200 decks would require a field length of $23{,}600_{10} \approx 60{,}000_{8}$.

Efficiency of the linkage editor may be improved by increasing the buffer size (PARAM(1)). For NASTRAN, PARAM(1) = 2080 is used. Additionally, one deck requires $16{,}000_{10}$ words of text storage (PARAM(6) = 16000). Consequently, for a link of 300 decks, the field length works out as

$$\text{field length}_{10} = 15000 + 3000 + 16000 + 6240 = 40240_{10} \approx 120000_{8} \ .$$

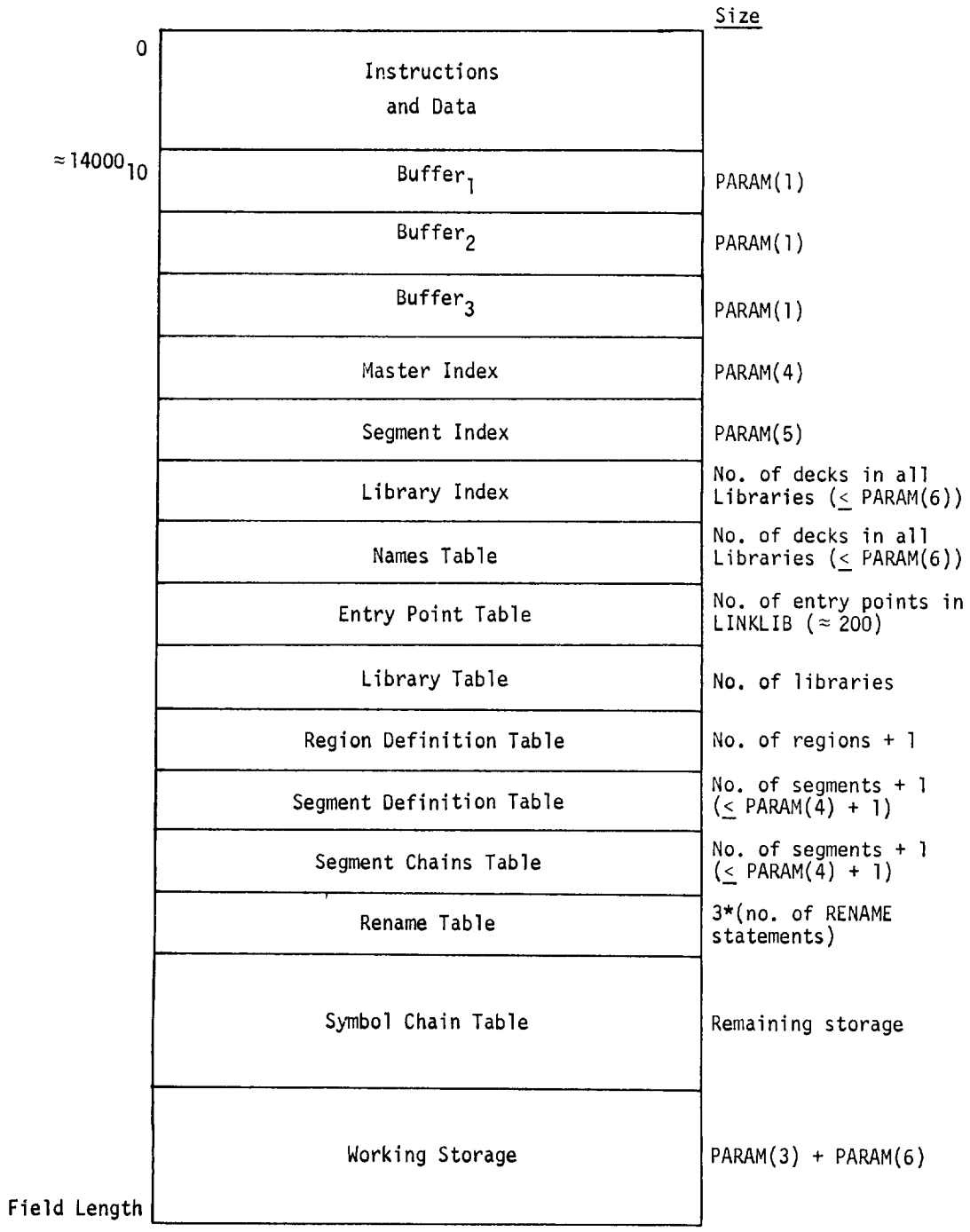


Figure 5. Layout of core storage for the linkage editor.

6.1 INTRODUCTION

Modifications and additions are continuously made to large programming systems. NASTRAN will not be an exception to this rule. Section 6.2 presents the FØRTRAN IV language restrictions that must be followed in order to produce equivalent object code across the computing machines on which NASTRAN operates. The remaining sections discuss areas of the program which experience, gained during program development, has shown to be those areas most subject to modifications and additions.

6.2 FØRTRAN IV LANGUAGE RESTRICTIONS

NASTRAN was developed in the FØRTRAN IV programming language to the greatest extent possible in order to simplify the task of conversion from the development machine, which, for the majority of program development, was the IBM 7094/7040(44) DCS, to third generation computing systems. The same FØRTRAN IV code can execute differently across computing machines. This fact is all but too well known to those who have labored through the task of conversion from one computing system to another. For this reason, modifications and additions to NASTRAN must be accomplished with FØRTRAN code that will produce equivalent object code across computing machines. The basic set of rules governing programming in FØRTRAN IV for NASTRAN is incorporated in the following manual in the IBM Systems Reference Library: IBM 7090/7094 IBSYS Operating System, Version 13, FØRTRAN IV Language, File No. 7090-25, Form C28-6390-3. The following is a list of exceptions to the rules set forth in this manual.

1. An integer constant may not be greater than $2^{31}-1$.
2. Subscripted variables should contain no more than 3 subscripts.
3. A reference to the first variable in a subscripted array must contain the subscript 1, e.g., A(1) = 0.0.
4. A CØNTINUE statement requires a FØRTRAN statement number.
5. The PAUSE statement is not to be used.
6. The NAMELIST statement is not to be used.
7. Implied DØ's in DATA statements are not allowed.
8. The last statement of a DØ loop may not be a logical IF statement.
9. BLØCK DATA subprograms may contain only type (e.g., REAL, INTEGER), DIMENSIØN, CØMMØN and DATA statements.
10. All Hollerith data should be defined in the form 4H.....
11. Do not use octal (Ø) in DATA or FØRMAT statements.
12. Specification statements should precede any executable statement.

13. The order of specification statements should be as follows:

    CØMPLEX

    DØUBLE PRECISIØN

    REAL

    INTEGER

    LØGICAL

    EXTERNAL

    DIMENSIØN

    CØMMØN

    EQUIVALENCE

    DATA

14. The variables in blank CØMMØN or a block of CØMMØN should be ordered as follows: complex, double precision, real, integer and logical.

15. Variables stored as single precision cannot be referenced as double precision variables (via the FØRTRAN EQUIVALENCE statement) because of the different internal word storage format for single and double precision words on the Univac 1108.

16. Caution must be exercised to insure that types (REAL, INTEGER, etc.) of FØRTRAN function values agree in the function subprogram and in the calling program. This agreement between types is necessary for machines (e.g., IBM S/360) on which REAL and INTEGER values of FØRTRAN functions are returned in different registers.

17. Do not attempt to extend the length of arrays through the EQUIVALENCE statement.

18. Caution must be exercised when using the EQUIVALENCE statement. One should not use the EQUIVALENCE statement to give different variable names to the same cell, since modern compilers, because of their optimization techniques, do not guarantee that the values of the equivalenced variables will be the same. Hence EQUIVALENCE should be used only between variables which have non-intersecting use spans in a program.

19. Nonstandard returns in a SUBROUTINE statement must immediately follow the left parenthesis which starts the names of the subroutine's arguments, e.g. SUBROUTINE XYZ (*,*,A,B) is the correct form; SUBROUTINE XYZ (*,A,*,B) is not acceptable.

20. There must be agreement with respect to the number of arguments and the type of each argument in the argument list of a calling program and the subprogram called.

21. For consistency with current NASTRAN practice, deck (or member) names for subroutines should agree with the primary entry point names. Deck names for Block Data subprograms should end with the characters "BD".

6.3 THE EXECUTIVE CONTROL DECK

The capabilities of the Executive Control Deck may be changed or increased by modifying existing control card functions or adding completely new card types. Executive control cards are processed within two modules of the Preface: XCSA, (Executive Control Section Analysis, section 4.2) and XGPI (Executive General Problem Initialization, section 4.7). Some cards are handled completely within XCSA, while others are only partially checked by XCSA and then passed to XGPI, via the Executive Control Table (section 2.4.2.5), for final processing.

To modify the content or function of an existing control card, first locate the proper section within module XCSA. The block of FØRTRAN statements related to the processing of each type of card is appropriately commented. Also, it can be determined from these statements whether part of the processing is being passed to module XGPI. The required modifications can then be made within XCSA and/or XGPI.

To add a new control card type to those currently acceptable, the following steps should be taken. First, add the card type name to the local FØRTRAN array ECTT (Executive Control Type Table) within module XCSA, and increase the table length parameter (LECTT) by three for each new entry. The three word entry consists of two BCD words (4 characters/word) for the card type mnemonic and a one word integer flag indicating whether the card type is to be optional (=0) or required (=1) within the NASTRAN Executive Control Deck. Second, add a statement number to the computed-go-to branch vector. This branch vector transfers the XCSA logic to the correct card processing section within the module. Third, create the processing code, and add it to the module. If additional processing must be performed in XGPI, the Control Table format should be modified and the necessary logic added to the XGPI module.

6.3.1 The NASTRAN Card

A facility is provided whereby the default values in /SYSTEM/, which are initialized by the system Block Data subprogram, SEMDBD, or subroutine BTSTRP, can be altered at execution time. The contents of /SYSTEM/ are described in section 2.4.1.8.

The card which provides this capability is called the NASTRAN card. If this card is used, it must be the first card of the data deck (i.e., the card must precede the Executive Control

Deck). The NASTRAN card is a free field card (similar to cards in the Executive Control and Case Control Decks). Its format is as follows:

NASTRAN keyword$_1$=value, keyword$_2$=value, ...

where the list of allowable keywords is as follows:

1. BUFFSIZE - Defines the number of words in a GINØ buffer. Note: fixed length records written by GINØ are of length BUFFSIZE - 3. This keyword changes the first word of /SYSTEM/.

2. MAXØPEN - Defines the maximum number of files that may be open at any one point in the program. This keyword changes the 30th word of /SYSTEM/.

3. CØNFIG - Defines the computer configuration for use in the timing equations in the matrix decomposition subroutines SDCØMP, DECØMP and CDCØMP. This keyword changes the 28th word of /SYSTEM/.

4. MAXFILES - Defines the maximum number of files to be placed in /XFIAT/ by GNFIAT. This keyword changes the 29th word of /SYSTEM/.

5. SYSTEM(I) - I refers to the Ith word of /SYSTEM/. This is a general form of altering any word in /SYSTEM/. Note that BUFFSIZE and SYSTEM(1) are equivalent, and MAXFILES and SYSTEM(29) are equivalent.

Examples of use of the NASTRAN card follow.

NASTRAN BUFFSIZE=878, SYSTEM(2)=3, MAXØPEN=10

The above card changes the 1st, 2nd and 30th words of /SYSTEM/. SYSTEM(2)=3 changes the system output unit from 6 to 3.

NASTRAN SYSTEM(4)=4, MAXFILES=21

The above card changes the 4th and 29th words of /SYSTEM/. SYSTEM(4)=4 changes the system input unit from 5 to 4 (which means that all subsequent data must be present on unit 4).

6.4 THE CASE CONTROL DECK

The Case Control Deck is processed by the IFP1 module, whose Module Functional Description can be found in section 4.3. A card can be added to the Case Control Deck by implementing the following steps.

1. Assignment of a Word in the CASECC Data Block (see section 2.3.1.1).

   If the card datum is to be passed on, a word must be assigned in CASECC. Several words are currently empty in CASECC. If more space in the fixed portion of CASECC is needed, modules which use CASECC to prepare data blocks for the Output File Processor (ØFP) module (e.g., SDR2, VDR, PLA3), will need to be updated since they are sensitive to the length of CASECC. Otherwise, just change the value of LENCC in /IFP1A/. Some Case Control Cards only change cells in /SYSTEM/ and do not need space in CASECC.

2. Addition (or change of) a Key Word.

   To add another key word simply lengthen /IFP1A/ by the number of new words, changing the IFPABD Block Data subprogram. The same procedure will allow you to change the spelling of a current keyword.

3. Identification of Card Type.

   In IFP1 there are approximately 30 logical IF statements in a row. Simply add another one modeled after the existing statements with the new key word.

4. Addition of Card Dependent Code.

   Add a small internal subroutine to process the new card. The simplest form of such an internal routine is to extract one integer from a card of the form SPC = 1. In this case set "IK" to the word assigned to the card in CASECC and transfer to the common code for this purpose. There are many examples of more complex cards under each card type. Changing these card dependent areas of code allows easy modification of existing card types.

5. Restart Implications

   See the subroutine description for IFP1B in the Module Function Description for IFP1 for a description of the restart functions of IFP1.

6.5 THE BULK DATA DECK

The module which processes the Bulk Data Deck is the Input File Processor (IFP), whose Module Functional Description can be found in section 4.5. There are two primary reasons for adding or modifying a bulk data card. First, a new structural element or some other item is to be added to the NASTRAN system. This will require additional code in several other modules besides the Input File Processor (IFP); however, only IFP changes will be discussed here. Second, an alternate form of user input is desired for an already existing item. For example, the SPC1 card is an alternate form of the SPC card. In some cases, an alternate form can be accommodated on the same card. An example of this technique is found on the SPØINT card. The advantage gained in this case is that changes to the NASTRAN system are isolated to the Input File Processor.

There are three major references for the programmer who desires to make modifications or additions to the Input File Processor. Section 2 of the User's Manual gives a functional description of each bulk data card. Section 2.3.2 of the Programmer's Manual describes the format of the output data blocks generated by the Input File Processor, and section 4.5 of the Programmer's Manual contains a description of the processing flow which occurs within IFP. Any programmer responsible for making changes to the Input File Processor would be well advised to select a card which is similar to the one he is changing or implementing and "follow it through" the code, using the three references described above to guide him.

In most cases, the work required to add a new card will amount to adding entries to already existing tables in the IFP Block Data subprograms. The detailed steps for adding a new card to the Input File Processor are listed below.

1. Add the card name and corresponding table data to the IFP data tables contained in Block Data Subprograms IFX1BD, IFX2BD, IFX3BD, IFX4BD, IFX5BD, IFX6BD, and if needed, IFX7BD. The meaning of these entries is discussed in section 4.5.7.

2. Add an entry in the IFP code to call one of the IFP secondary routines IFS1P, IFS2P, IFS3P, or IFS4P, wherein the card dependent processing takes place, and add the necessary card dependent code to the appropriate secondary routine.

3. If the new card is to be used in conjunction with a Rigid Format, appropriate entries must be made in the restart bit tables in /IFXO/ and in the Rigid Format tables. See section 1.10 for details.

6.6 RIGID FORMATS

The following steps will allow the addition of a new NASTRAN Rigid Format.

1. Compile and test the DMAP sequence throughly by running problems on it using APPRØACH = DMAP or by using the ALTER feature with an existing Rigid Format.

2. All Rigid Formats must be classified FØRCE or DISPLACEMENT. A call to a subroutine which stores the new Rigid Format must be added to subroutine XRGDFM of module XCSA (see section 4.2). The subroutines which contain current Displacement Rigid Formats are called LDi, i = 01, 02, ..., 12. Dummy calls are already setup for Displacement Rigid Formats LD45-LD50. They correspond to solutions 13 through 19. By choosing to use solutions 13 through 19 and creating the appropriate LDi routine, this step can be skipped.

3. An LDi routine must be written. The LDi routine must write the DMAP sequence, the Decision Tables, the File Name Table, and the Card Name Table on the NPTP. The format is as follows:

Record 1:

This record contains the coded DMAP sequence, 4 characters per word. Note that every DMAP instruction must end with the character: $. In the existing LDi programs the data for this record are stored in the RD array.

Record 2:

Number of Words	Contents
1	Number of DMAP instructions (NDMAP).
1	Number of words in the decision table for each DMAP instruction (NBIT), $1 \leq NBIT \leq 5$.
NBIT*DMAP	Decision table for each instruction with the entries for instructions not in this subset set to zero. The words cannot be all zero. The zeroing of the subset entries can be accomplished by a call to XSBSET. (See XCSA MFD write-up). In the existing LDi programs, this table is stored in the IS1 array.
1	Number of entries in the File Name Table (NFILE). This number may be zero if no File Name Table is desired.

Number of Words	Contents
3*NFILE	File Name Table - Each entry consists of 2 BCD words giving the data block name of any data block in the Rigid Format and one integer giving its bit position in the Decision Table. In the existing LDi programs, the File Name Table is stored in the JNM array.
1	Number of entries in the Card Name Table (NCARD). This number may be zero if no Card Name Table is desired.
3*NCARD	Card Name Table - Each entry consists of 2 BCD words giving the card Name of any NASTRAN Data Card and one integer giving its bit position in the Decision Table. In the existing LDi programs, the Card Name Table is stored in the INM array.

To modify an existing rigid format the appropriate tables (Decision, Card Name, File Name or DMAP) should be changed. Their locations in the existing LDi routines are detailed above.

6.7 FUNCTIONAL MODULES

A functional module communicates with the NASTRAN Executive System, and hence indirectly with other functional modules, only through its input data blocks, its output data blocks and its DMAP parameters. Hence a modification to a functional module which disturbs neither its output data blocks nor its DMAP parameters can be made without changing any other functional modules. If a modification to a functional module affects its output data blocks or its DMAP parameters, it must be determined (by referring to section 4, Module Functional Descriptions, of the Programmer's Manual and/or section 3, Rigid Formats, of the User's Manual) what modules use these output data blocks and DMAP parameters as input. If these modules are numerous, or if the changes to them are very extensive, it may be more profitable to write a new module(s) to accomplish the task at hand.

To add a functional module to the system three changes must be made: 1) update the Module Properties List (MPL), the Executive table which contains the (DMAP) name of the module, the number of input data blocks, the number of output data blocks, the number of scratch data blocks and the DMAP parameter list (see the description of the MPL in section 2.4 for more details); 2) update the Link Specification Table (LNKSPC), the Executive table which contains the (DMAP) name of the module, the module's entry point name and the link residence keys for the four machine types (see the description of the LNKSPC table in section 2.4 for more details); and 3) update one or more link driver routines, XSEMi, (see section 3.3.7) so that the module is called from one or more links. Steps 2 and 3 have been automated within the NASTRAN system. These automated procedures are described in sections 6.11.3.1 and 6.11.3.2 respectively.

6.8 STRUCTURAL ELEMENTS

Code which is structural element dependent is found in the following modules: IFP, IFP3 (conical shell element only), GP1, GP2, GP3, TA1, PLTSET, SMA1, SMA2, SSG1, SDR2, DSMG1, PLA1, PLA3, PLA4, and ØFP. To modify element dependent code in any of these modules, the corresponding Module Functional Description in section 4 and the mathematical descriptions of the elements at the end of section 4 should be consulted. To add an element to the NASTRAN system the following changes are required to the above referenced modules.

6.8.1 IFP (Input File Processor)

An element connection (e.g., CRØD) bulk data card will have to be designed and added to the set of admissible NASTRAN bulk data cards. An element property card (e.g., PRØD) may have to be added. See section 6.5 for details on additions to the Bulk Data Deck.

6.8.2 GP1, GP2, GP3, and TA1 (Geometry Processor and Table Assembler Modules)

Processing of element data in GP1, GP2, GP3, and TA1 is entirely controlled by descriptive information contained in /GPTA1/ (see section 2.5.2.1). If a new element is to be added to NASTRAN, the following changes are made to /GPTA1/:

WORD	CHANGE
1	Increase the number of entries by 1. Call the new number of entries n.
2	Increase the pointer to the last element by the number of words per entry (currently 13).
3	No change
4 + (n-1)*13 5 + (n-1)*13	Include the new element name.
6 + (n-1)*13	Assign a new element type identification number which is unique.
7 + (n-1)*13 : 16 + (n-1)*13	Complete items in accordance with definitions in section 2.5.2.1.

Notes:

1. The new entry should be inserted so that the alphabetical sort on words 1 and 2 of each entry in the table is maintained.

2. The new element type identification number will be used by any module reading the ECPT, EST, ECPTNL, ECPTNL1, ESTNL or ESTNL1 data blocks.

6.8.3 PLTSET (Plot Set Definition Processor)

Processing of element data in PLTSET is controlled by descriptive information in /EDES/, which is initialized in the EDESBD Block Data subprogram (see section 4.23.9.2). If the new element to be added is to be plotted, the following changes are to be made to /EDES/.

1. Increase the first word of /EDES/, the number of element types defined in this table, by one.

2. Add the two word BCD name of the element to the ENAM array in /EDES/. The name of the element here is defined to be the connection bulk data card mnemonic with the leading character "C" deleted. For example, the name of the element defined by a CRØD connection card would be "RØD".

3. Add the two word element code defined for subroutine LØCATE to the array ECØD in /EDES/, so that the connection card record in the ECT data block may be found.

4. Update the EWRD, ENGP and ESYM arrays in /EDES/ (see section 4.23.9.2).

6.8.4 SMA1 (Structural Matrix Assembler - Phase 1)

In order to add an element the following changes must be made in module SMA1.

1. The FØRTRAN arrays IØVRLY and NWØRDS in /SMA1CL/ must be updated in Block Data subprogram SMA1BD.

2. The computed-go-to statement in subroutine SMA1A which performs a branch on element identification number must be changed, and a call to the new element routine must be inserted following this computed-go-to statement.

3. The element subroutine must "insert" only the 6 by 6 submatrices of the element stiffness matrix into the submatrix of $[K_{gg}]$ or $[K^4_{gg}]$ in core corresponding to the current pivot point, NPVT, which is contained in /SMA1CL/. The "insertion" routine is subroutine SMA1B.

6.8.5 SMA2 (Structural Matrix Assembler - Phase 2)

The changes outlined above for module SMA1 apply equally to module SMA2 with the following name changes: 1) replace /SMA1CL/ with /SMA2CL/; 2) replace SMA1BD with SMA2BD; 3) replace SMA1A with SMA2A; 4) replace SMA1B with SMA2B; and 5) replace $[K_{gg}]$ with either $[M_{gg}]$ or $[B_{gg}]$.

6.8.6 SSG1 (Static Solution Generator - Phase 1)

Subroutine EDTL of module SSG1 is a driver routine which: 1) stores the thermal field, 2) initializes the MAT routine and 3) processes the EST data block one element type at a time. Several changes must be made to EDTL to add an element.

1. An additional entry must be made to the local FØRTRAN array NECPT. The entries are the number of words per element in the EST.

2. A computed-go-to on ELTYPE must be changed to point to the new element dependent code.

3. Element dependent code must be added to: 1) read the EST element entry into /TRIMEX/, 2) look up the temperature at each element grid point (subroutine FGPTT), and 3) call the element routine giving these temperatures as well as the beginning of the load vector array.

In addition, an element thermal routine must be written to compute the equivalent loads. The results of these computations should be accumulatively added to the load vector array. Several general purpose subroutines are available for these computations: MAT for finding material properties; MPYL and MPYLT for in-core matrix multiplication; BASGLB and GLBBAS for coordinate system transformations. The descriptions of these routines can be found in section 3 or in the "subroutines" section, section 4.41.11, of the Module Functional Description for SSG1.

6.8.7 SDR2 (Stress Data Recovery - Phase 2)

If an element is to be added, the following changes to the SDR2 module must be accomplished.

1. Create two computation routines: SXXXX1, to generate element stress matrices; and SXXXX2, to perform final stress and force computation. Model the above two routines after element routines now in the NASTRAN program, e.g., SRØD1 and SRØD2. The "XXXX" in the names SXXXX1 and SXXXX2 should be 4 letters of the element name.

2. Block Data changes to SDR2BD.

   a) Increase the dimension of ELEM2 by 9 words, and initialize these words with the following:

Word	Description
1, 2	Element name (BCD - 4 characters/word)
3	Number of words for the element entry in the EST data block.
4	Number of grid points associated with the element.
5	Number of words output by SXXXX1.
6	Number of stress words output by SXXXX2.
7	Number of force words output by SXXXX2.
8	Pointer to the first word of complex stress output string in the CØMPLX array of subroutine SDR2E.
9	Pointer to the first word of complex force output string in the CØMPLX array of subroutine SDR2E.

   b) Increase data count of NELEM by 1, and increase data count of LAST by 9.

3. The following change must be made to subroutine SDR2B.

   a) Add an address to the computed-go-to branch vector, and add the call to SXXXX1.

4. The following changes must be made to subroutine SDR2E.

   a) Add an address to the computed-go-to branch vector, and add the call to SXXXX2.

   b) Increase the dimension of the CØMPLX array, and add a pointer string for stresses and forces. If complex stresses and forces are not permitted, set the two CØMPLX pointers in the Block Data subprogram SDR2BD to 0.

6.8.8 DSMG1 (Differential Stiffness Matrix Generator - Phase 1)

Changes must be made to module DSMG1 if either a new element is to have contributions to the differential stiffness matrix or a current element which currently has no contribution to the differential stiffness matrix is to have contributions.

The following changes must be made in subroutine DS1.

1. The local arrays DSARY, NWØRDS and NGPTS must each be updated. The array DSARY is defined as follows: DSARY(I) = 1 if the I^{th} element type is an element for which

differential stiffness is defined; otherwise, DSARY(I) = 0. NWØRDS(I) is the number of words of an ECPT data block entry for the I^{th} element, and NGPTS(I) is the number of grid points associated with the I^{th} element.

2. When appending the components of the displacement vector to the ECPT element entry, either the three translational components of displacement at each grid point or all six components of displacement at each grid point can be appended.

The following changes must be made in Block Data subprogram DS1ABD and in subroutine DS1A.

1. The FØRTRAN arrays IØVRLY and NWØRDS (the number of words to be read from the ECPTDS scratch file), which reside in /DS1AAA/, must be updated. If another element routine overlay segment is constructed, NLINKS must be changed. These changes are to be made in DS1ABD.

2. The computed-go-to statement in subroutine DS1A which performs a branch on element identification number must be changed, and a call to the new element routine must be inserted.

3. The element subroutine must "insert" only the 6 by 6 submatrices of the element differential stiffness matrix into the submatrix of $[K^{d}_{gg}]$ in core corresponding to the current pivot point, NPVT, which is contained in /DS1AAA/. The "insertion" routine is subroutine DS1B.

6.8.9 PLA1 (Piecewise Linear Analysis - Phase 1)

The following changes must be made to module PLA1 if either the added element is to be admissible to the set of elements for which Piecewise Linear Analysis is defined or a current element is to change its status to an element for which Piecewise Linear Analysis is defined.

1. The computed-go-to (on element identification number), which reflects whether an element is in the set of elements for which Piecewise Linear Analysis is defined, must be altered.

2. Element dependent code which initializes stress information appended to the ECPT and EST data blocks must be added. The commented code in PLA1 along with the descriptions in section 2 for the ECPTNL and ESTNL data blocks serve as models for the addition of this code.

3. The local array, PLAARY, must be updated. PLAARY(I) = 1 if the I^{th} element type is an element for which Piecewise Linear Analysis is defined, and PLAARY(I) = 0 otherwise.

6.8.10 PLA3 (Piecewise Linear Analysis - Phase 3)

The following changes must be made to module PLA3 if either the added element is to be admissible to the set of elements for which Piecewise Linear Analysis is defined or a current element is to change its status to an element for which Piecewise Linear Analysis is defined.

The following changes must be made to subroutine PLA31.

1. The local arrays ESTWDS, which defines the number of words to be read from the ESTNL data block for each element, and NGPTS, which defines the number of grid points for each element, must be updated.

2. When appending the components of the displacement vector to the ESTNL element entry, either the three translational components of displacement at each grid point or all six components of displacement at each grid point can be appended.

The following changes must be made to subroutine PLA32.

1. The array ESTWDS, as defined above, must be updated. The array NSTWDS, which defines the number of words per element written on the ØNLES data block, must be updated. The array NWDSP2, which defines the number of words per entry in the scratch data block ESTNLS, must be updated.

2. The computed-go-to statement which performs a branch on element identification number must be changed, and a call to the new element routine must be inserted following this computed-go-to statement.

6.8.11 PLA4 (Piecewise Linear Analysis - Phase 4)

The following changes must be made to module PLA4 if either the added element is to be admissible to the set of elements for which Piecewise Linear Analysis is defined or a current element is to change its status to an element for which Piecewise Linear Analysis is defined.

The following changes must be made to subroutine PLA41.

1. The local arrays NWØRDS and NGPTS must be updated. NWØRDS(I) is the number of words to be read from the ECPTNL data block, for the I^{th} element, and NGPTS(I) is the number of grid points associated with the I^{th} element.

2. When appending the components of the displacement vector to the ECPTNL element entry,

either the three translational components of displacement at each grid point or all six components of displacement at each grid point can be appended.

The following changes must be made to Block Data subprogram PLA4BD and subroutine PLA42.

1. The variables IØVRLY, NWØRDS, and possibly NLINKS, are to be changed in PLA4BD.

2. In PLA42, the local array NWDSP2, which defines the number of words per element entry to be written on the ECPTNL1 data block, must be updated.

3. The computed-go-to statement in PLA42 which performs a branch on element identification number must be changed, and a call to the new element routine must be inserted following this computed-go-to statement.

4. The element routine must "insert" only the 6 by 6 submatrices of the element stiffness matrix into the submatrix of $[K_{gg}^{n\ell}]$ in core corresponding to the current pivot point, NPVT, which is in /PLA42C/.

6.8.12 ØFP (Output File Processor)

ØFP will automatically print, using the TABPT routine, any data not recognized, and this includes undefined (new) element stress and force output. If standard NASTRAN formatting is desired, the NASTRAN systems programmer should implement additional heading formats to ØFP1A and pointers in the Block Data subprograms ØFP1BD, ØFP2BD, ØFP4BD, and ØFP5BD. This data may be selected by noting the data for currently operating element outputs. Reference the Module Functional Description for ØFP and the data block descriptions in section 2 for any stress or force output data block, e.g., ØES1, ØEF1

6.9 PRINTED OUTPUT

The majority of the NASTRAN data scheduled for output are input to the Output File Processor (ØFP) module. This module performs the actual formatting and outputting of the data.

To implement additional output capability in the ØFP, the following ground rules should be observed in designing a data block intended to be input to ØFP for output to the system printer or punch unit.

1. The data block's name, for consistency, should begin with an "Ø".

2. There should be one or more repeating record pairs, where the record pair is of the following form:

WORD	MODE	DESCRIPTION
1	Available for Data	
2	I	Major ID
3	I	Minor ID
4	I	Subcase ID
5		
.		
.	Available for Data	
.		
9		
10	I	Number of words per entry
11		
.		
.	Available for Data	
.		
50		
51	BCD	
.	.	32 Words Title
.	.	32 Words Subtitle
.	.	32 Words Label
146	BCD	

Identification Record

(An odd-numbered Record)

Repeating data entries of the length specified in the preceding record. This record may be null but should exist.

Data Record (An even-numbered Record)

3. The Major ID is formulated depending on the data type as follows.

   a. For a new data type an integer should be selected which equals I, where I is a value one greater than the current number of accepted data types within ØFP.

   b. The Major ID then depends on the data classification

$$\text{Major ID} \begin{cases} \text{I} = \text{SØRT1 - Real} \\ 1000 + \text{I} = \text{SØRT1 - Complex} \\ 2000 + \text{I} = \text{SØRT2 - Real} \\ 3000 + \text{I} = \text{SØRT2 - Complex} \end{cases}$$

4. The Minor ID is optional to distinguish subclasses of the Major ID.

With the data block format outlined above, the NASTRAN systems programmer may implement the new data formats within the current ØFP procedure by modeling from any current data type now handled by ØFP. The current ØFP has dynamic formatting for the outputting of entries of the Data Record. However, if the programmer is not familiar with the five level pointer procedure used for the dynamic formatting, it is recommended that he implement logic to explicitly output the Data Record.

ØFP will, without modification, output new data blocks via the table print routine, TABPT.

6.10 PLOTTER OUTPUT

Plotted output in NASTRAN is restricted to three modules: 1) the Structural Plotter (PLØT - see section 4.24); 2) the XY Plotter (XYPLØT - see section 4.69); and 3) the Matrix Plotter (SEEMAT - see section 4.74). Each of these modules uses the NASTRAN plotter software package of utility routines, each of which is individually documented in section 3.4.

6.10.1 Changes to the Plotter Software

In order to add a new plotter to the NASTRAN plotter software, the following must be done: 1) up to four (4) new subroutines (AXISi, LINEi, TYPEi, WPLTi) must be written, where "i" is the internal index of the new plotter (see section 3.1 for the current values of i); and, 2) changes and/or additions must be made to ten (10) existing subroutines (AXIS, FNDPLT, LINE, PLØTBD, SELCAM, SKPFRM, SYMBØL, TIPE, TYPFLT, TYPINT).

Initially, it should be decided which of the four possible new subroutines will be needed for the new plotter. For the most part, three will be the minimum: LINEi, TYPEi, WPLTi. If the new plotter has no typing capability, an existing subroutine, DRWCHR, will be used in place of TYPEi. It is strongly recommended that the calling sequences of any new subroutines be the same as those of their existing counterparts. None of the four possible new subroutines should present much difficulty to the programmer if he uses their existing counterparts as models.

Subroutine AXISi need only be written if the following two conditions are true on the new plotter: 1) lines must be drawn generally as a series of short lines; and 2) there does exist a special plotter instruction permitting the drawing of a horizontal or vertical line without representing it as a series of short lines. Both of these conditions do exist for plotter 3 and may exist for plotter 10. If either one of these two conditions is not true for the new plotter, then LINE should be used in place of AXISi.

Subroutine LINEi is always required. If the new plotter must be put into the line drawing mode before any lines can be drawn, LINE1, LINE2, LINE9 and LINE10 should be used as models. If the new plotter is an incremental plotter, LINE4 should be used as the model. Otherwise, use LINE3 as the model. In addition, LINE1, LINE2 and LINE3 represent lines as series of short lines; LINE9 and LINE10 use only one plot command to draw a line of any length.

Subroutine TYPEi should be written if there is a typing capability on the new plotter. If the new plotter must be put into the typing mode before any lines can be drawn, TYPE1, TYPE2, TYPE9 and TYPE10 should be used as models. Otherwise, use TYPE3 as the model. All characters should be vertically oriented and miniature in size. Each of the model subroutines has three important symbols initialized in a DATA statement: LSTCHR, NCHR and CHAR. LSTCHR is the index of the last legitimate character in the CHAR94 table (see section 2.5) available on the new plotter. NCHR is the number of characters for which the character code on the new plotter differs from the corresponding character code in the CHAR94 table. CHAR is a list of NCHR pairs of indices into the CHAR94 table: $CHAR_{1,j}$ is the index value of the character code which will produce the wrong typed character on the plotter, and $CHAR_{2,j}$ is the index value of the character code which will produce the correct typed character. One further note: some plotters have both a "typewriter" and "single-character typing" mode. The existing TYPEi subroutines use only the "single-character typing" mode. It is strongly recommended that this policy be continued for all new plotters.

Subroutine WPLTi is almost entirely dependent upon the structure of the plot commands for the new plotter. The existing WPLTi subroutines will help the programmer only insofar as the overall logic is concerned. Once the plot command is constructed, it is written using subroutine SWRITE. Some plotters also require special information at the beginning and/or end of each record on the plot tape. If this is the case with the new plotter, WPLT3, WPLT4, WPLT9 and WPLT10 contain the logic necessary to recognize the beginning and/or end of a plot tape record. In addition, all the existing WPLTi subroutines, except for WPLT4, generate fixed-length plot commands. This results in a simpler subroutine and also adapts easier to the computer-independent characteristic of NASTRAN. It is therefore recommended that this policy be continued, if at all possible, for all new plotters.

Of the ten (10) subroutines to which changes and/or additions must be made, changes in six of the ten subroutines are dependent upon a) the internal index of the new plotter; and b) which of the four possible new subroutines have been written. These six (6) subroutines are: AXIS, LINE, SYMBØL, TIPE, TYPFLT and TYPINT. In each there are one or two computed-go-to statements based upon the internal index of the new plotter which will have to be enlarged. Then corresponding statements calling the new subroutines (AXISi, LINEi, or TYPEi) will have to be added. If an AXISi subroutine was not written for the new plotter, subroutine LINE should be called

from subroutine AXIS instead of calling AXISi. If a TYPEi subroutine was not written for the new plotter, then subroutine DRWCHR should be called from subroutines SYMBØL, TIPE, TYPFLT and TYPINT instead of calling TYPEi.

Subroutine FNDPLT relates the external plotter and model names with its internal plotter and model indices. There is a table within this subroutine which reflects this relationship. The programmer need only append the external name of the new plotter and/or an additional model name to an existing plotter, together with the corresponding internal plotter and model indices. In addition, the value of the variable NPLTRS will have to be incremented by one whenever a new external plotter name is added.

Subroutine PLØTBD is a Block Data subprogram. Both the PLTDAT and SYMBLS tables (see section 2.5) must be expanded to reflect the addition of the new plotter. A new 20-word section must be appended to the PLTDAT table reflecting the physical characteristics of the new plotter. The values in this new section must correspond on a one-to-one basis with the values defined for the existing plotters. In addition, the buffer size defined for the new plotter must be a multiple of 60 characters if computer independency is to be preserved. A new 20-word section must be appended to the SYMBLS table. The first "NSYM" values in this new section are indices into the CHAR94 or CHRDRW tables (see section 2.5). These indices represent the special symbols used by the SYMBØL subroutine for the new plotter.

A new section must be added to subroutine SELCAM. The function of the new section varies depending upon the requirements of the new plotter. If the new plotter is a table plotter, a plot command should be generated to stop the plotter so that the operator can prepare for the next plot. If the new plotter is a microfilm plotter, the requested camera must be selected. Some plotters require a special "header" record at the beginning of each plot. If so, it must be generated in this subroutine. Again, the function of this subroutine varies depending upon the requirements of the new plotter, and as a result, the added section is very likely to be entirely plotter-dependent. In addition, there are two computed-go-to statements (based upon the internal index of the new plotter) which must be enlarged to reflect the new plotter.

The last subroutine to which changes and/or additions must be made is SKPFRM. Its function is to skip a variable number of frames on a microfilm plotter, or to skip over the current plot on a drum plotter. If the new plotter is neither a microfilm nor a drum plotter, no new section

is needed. In either case, there is a computed-go-to statement (based upon the internal index of the new plotter) which must be enlarged to reflect the new plotter. If the new plotter is a microfilm plotter, the commands necessary to skip the frame(s) must be added to the subroutine (e.g., plotters 3 and 9). If the new plotter is a drum plotter, then the commands necessary to skip over the current plot must be added to the subroutine (e.g., plotter 4). As with subroutine SELCAM, any added section is very likely going to be plotter-dependent.

Finally, an important word of caution to the programmer must be stated: if the computer and installation independence of the NASTRAN plotter software package is to be maintained, no existing software provided for his computer or installation ought to be used to accomplish the task of adding a new plotter to the NASTRAN plotter software package. This restriction prevents the direct usage of on-line plotters. NASTRAN does provide however for the indirect usage of on-line plotters as described in section 6.10.6.

6.10.2 Changes to the PLØT Module, the Structural Plotter

In order to add a new plotter to the NASTRAN structural plotter module, PLØT, only one subroutine need be altered: PLTØPR. This subroutine generates messages to the plotter operator. Probably the only message which should be added is one to identify the plotter for which the structural plots are being generated. There is a repertoire of other messages in this subroutine which should be general enough to apply to any plotter type. If additional messages are needed, the programmer must remember that the plotter operator is not normally a "programmer type". Hence he will be more apt to respond to messages written in his language than to messages written in programming terminology. In addition, there are several computed-go-to statements based upon the internal index of the new plotter. These will have to be enlarged to reflect the new plotter.

6.10.3 Changes to the XYPLØT Module, the XY Plotter

No changes are required within the XYPLØT module when adding a new plotter. However, a minor change is required in subroutine IFP1XY of the IFP module if and only if the new plotter uses the NASTRAN BCD plot tape, PLT1. If this is the case, the new internal plotter number which is generated by subroutine FNDPLT of the NASTRAN plotter software should be added to the FØRTRAN logical IF statement directly after the following comment in subroutine IFP1XY:

```
C
C  IF MØRE BCD PLØTTERS ARE ADDED EXPAND THE FØLLØWING TEST
C
```

6.10.4 Changes to the SEEMAT Module, the Matrix Plotter

No changes to the SEEMAT module are necessary when a new plotter is to be added since the plotter and model names are communicated to the module through the DMAP calling sequence (see section 4.74).

6.10.5 Use of the NASTRAN Plotter Software in a New Module

There exists in NASTRAN a self-contained computer-independent environment for creating plots on a large number of plotters. Currently three modules use this environment: 1) PLØT, the Structural Plotter; 2) XYPLØT, the XY Plotter; and 3) SEEMAT, the Matrix Plotter. In order for another module to generate plots in this same environment, it is essential that the module writer understand this environment.

The NASTRAN plotting environment can be understood most easily if viewed as being composed of five parts: 1) parameter definitions; 2) parameter initialization; 3) plot initiation; 4) plot creation; and 5) plot termination.

There is one DMAP parameter (passed through blank common) and two named common blocks in the parameter definition section of the NASTRAN plotting environment. The DMAP parameter is PLTNUM, the plot number of the last plot created, and is both an input and output integer value. The two named common blocks, with their symbolic contents, are as follows:

```
CØMMØN/XXPARM/PBFSIZ,MEDIUM,BFRAMS,SKIP(4),XPAPSZ,YPAPSZ
```

where:

PBFSIZ = size of the plot tape buffer (integer)

MEDIUM = medium output request (integer, default value = 2)

$$= \begin{cases} 1 = \text{film only} \\ 2 = \text{paper only} \\ 3 = \text{both} \end{cases}$$

BFRAMS = number of blank frames (on film only) between plots (integer, default value = 1)

XPAPSZ = width of the paper (inches, for table plotters only) to be used (real, default value = 8.5 inches)

YPAPSZ = height of the paper (inches, for table plotters only) to be used (real, default value = 11.0 inches)

```
CØMMØN/PLTDAT/MØDEL,PLØTER,REGIØN(4),AXMAX,AYMAX,XEDGE,YEDGE,SKIP(10),PCNTIN,CNTCHX,CNTCHY,
      SKIP(4),PLTYPE,PLTAPE,SKIP(2),FCNTIN
```

where:

MØDEL = model index of the plotter to be used (integer, default value = 1)

PLØTER = plotter index of the plotter to be used (integer, default value = 3 - the SC 4020 microfilm plotter)

REGIØN = the region of the plotting surface (x_{min}, y_{min}, x_{max}, y_{max}) in which a plot is being created (real)

AXMAX = width of the plotting surface less any borders (real)

AYMAX = height of the plotting surface less any borders (real)

XEDGE = width of the border on the left and right side of the plotting surface (real)

YEDGE = height of the border on the top and bottom of the plotting surface (real)

PCNTIN = number of counts (plotter units) per inch of paper (real)

CNTCHX = the width (includes horizontal spacing between characters) of each printed or drawn character (real)

CNTCHY = the height (includes vertical spacing between characters) of each printed or drawn character (real)

PLTYPE = plotter type (integer)

$$= \begin{cases} \pm 1 = \text{microfilm plotter} \\ \pm 2 = \text{table plotter} \\ \pm 3 = \text{drum plotter} \end{cases}$$

$$= \begin{cases} <0 \text{ if no typing capability exists on the plotter.} \\ >0 \text{ if typing capability does exist on the plotter.} \end{cases}$$

PLTAPE = GINØ file name of the plot tape (BCD)

$$= \begin{cases} \text{PLT1 for an even parity plot tape} \\ \text{PLT2 for an odd parity plot tape} \end{cases}$$

FCNTIN = number of counts (plotter units) per inch of film (real, FCNTIN = PCNTIN if the plotter is not a microfilm plotter)

The usage and initialization of each of the variables listed above are detailed in the following descriptions of the four remaining sections of the NASTRAN plotting environment (see section 2.5 for a more detailed description of /XXPARM/ and /PLTDAT/).

The parameter initialization section of the NASTRAN plotting environment involves initializing all the variables listed above for the /XXPARM/ and /PLTDAT/ named common blocks, with the exception of MEDIUM and BFRAMS. First, the plotter and model indices (PLØTER, MØDEL) of the plotter to be used must be set by using subroutine FNDPLT:

```
CALL FNDPLT (PLTTER,MØDELN,PLTID,MØDID)
```

where:

PLTTER = plotter index of the plotter specified in PLTID and MØDID (integer, output)

MØDELN = model index of the plotter specified in PLTID and MØDID (integer, output)

PLTID(2) = 8 character name (4 characters per word, left-adjusted, with all remaining characters blank) of the plotter of interest (BCD, input).

MØDID(2) = model identification of the plotter of interest. $MØDID_i$ may either be numeric or BCD, depending upon the way in which various models are identified for the plotter of interest. If $MØDID_i$ is BCD, it must be 4 characters, left-adjusted, with all remaining characters blank. (Integer and/or BCD, input and output).

See section 3.1 for a list of current plotter and model names.

Subroutine FNDPLT will attempt to match PLTID with an internal list of plotters available in the NASTRAN plotting environment. If no match is found, FNDPLT sets both PLTTER and MØDELN = 0 and immediately returns. Otherwise, FNDPLT will attempt to match MØDID with an internal list of models for the plotter of interest. If no match is found, a default model for the plotter is stored in MØDID. Then the corresponding plotter and model indices are stored in PLTTER and MØDELN. Generally, PLTID and MØDID will be provided by the NASTRAN user. It is suggested that the module writer set MØDID(1) and MØDID(2) equal to zero before inserting the user supplied values for MØDID. This provides the user with the capability of defaulting to a standard default model for his specified plotter by supplying no values for MØDID or by supplying a value only for MØDID(1).

Having called subroutine FNDPLT to determine the plotter and model indices, the module writer should then set PLØTER and MØDEL equal to PLTTER and MØDELN, respectively:

```
PLØTER = PLTTER

MODEL = MØDELN
```

Next, the remainder of the /PLTDAT/ named common block must be initialized. Subroutine PLTSET must be used to do this. PLTSET sets all the variables in /PLTDAT/, except MØDEL and PLØTER, to values dependent upon the plotter index (PLØTER). In addition, if the plotter is a table plotter, AXMAX and AYMAX are also functions of the paper size (XPAPSZ, YPAPSZ). For this reason, if the current or default paper size is to be changed, it must be done before calling PLTSET. Finally REGIØN is initialized in PLTSET as follows:

```
REGIØN(1) = 0.
REGIØN(2) = 0.
REGIØN(3) = AXMAX
REGIØN(4) = AYMAX
```

and the plot tape buffer size (PBFSIZ) in the /XXPARM/ named common block is initialized.

This completes the parameter initialization section of the NASTRAN plotting environment. The next step is plot initiation. Once this step is initiated, none of the parameters in either the /XXPARM/ or /PLTDAT/ named common blocks may be changed, except for MEDIUM, BFRAMS and REGIØN, without repeating the parameter initialization step.

In the plot initiation step, the module writer must first ensure that the plot tape (PLTAPE) is indeed a physical tape. This can be done by using logical function TAPBIT. If the function result is .FALSE., the plot tape is not a physical tape and hence no plotting should be attempted by the module:

```
IF (.NØT.TAPBIT(PLTAPE)) "no physical tape setup"
```

Having verified the existence of a physical plot tape, an array of PBFSIZ words must be provided for the NASTRAN plotting software as follows:

```
CALL SØPEN ($n,PLTAPE,BUFFER,PBFSIZ)
```

where:

n = FØRTRAN statement number to which SØPEN is to return if PLTAPE has not been correctly initialized.

BUFFER = an array of PBFSIZ full words.

Next, the plotter must be started. However, before doing this, the plot number (PLTNUM) should be incremented by one. In addition, this is the module writer's last opportunity to change the medium request (MEDIUM) and the number of blank frames between plots (BFRAMS). To start the plotter, the module writer must call subroutine STPLØT:

CALL STPLØT (PLTNUM)

Subroutine STPLØT will select the proper medium (if appropriate), generate an identification plot (if necessary), and insert the specified number of blank frames (on film only).

Having initiated a plot, the module writer must then create the plot. In the plot creation section of the NASTRAN plotting environment, there are seven (7) subroutines provided for various tasks: AXIS, LINE, PRINT, SYMBØL, TIPE, TYPFLT, TYPINT. An explanation of the calling sequence and purpose of each of these subroutines exists in section 3. It is essential that the module writer familiarize himself with the calling sequences and purposes of these subroutines. In addition, it is just as essential that he also understand a certain amount of the philosophy which exists in these subroutines as a class. What follows is an attempt to explain this philosophy, together with pertinent suggestions.

There are three operating modes defined in these seven subroutines: 1) axis mode; 2) straight line mode; and 3) typing mode. These three modes are totally independent of each other and are mutually self-exclusive. What this actually implies is that only one mode can be active at any one point in time. Subroutine AXIS operates in the axis mode; subroutine LINE operates in the straight line mode; and subroutines PRINT, SYMBØL, TIPE, TYPFLT and TYPINT all operate in the typing mode.

Each of these subroutines has a common argument, which is always the last argument in each of the calling sequences. This argument (ØPT) is used to initiate a mode (ØPT=-1), operate within a mode (ØPT=0), or terminate a logical subset of plot commands within a mode (ØPT=+1). Only while operating within a mode (ØPT=0) do any of the other arguments in a calling sequence have any meaning. For this reason, it is usually a good practice to use zeros for the other arguments when initiating a mode or when terminating a logical subset of plot commands.

It is strongly recommended that the set of all commands used to create a plot be grouped into logical subsets of commands, with each subset operating in only one of the three possible modes.

This does not mean that all axes or all lines or all typing be included in one subset of commands. In many cases it is more logical to create several subsets of axis commands or straight line commands. The module writer must call one of the subroutines which operates within the same mode, with ØPT=-1, so that the mode will be properly initiated. It is then recommended that, following such a subset of commands, the module writer again call one of the subroutines which operates within the same mode, with ØPT=+1, to terminate the subset of plot commands. An example of these recommendations is as follows:

```
CALL TIPE (0,0,0,0,0,-1)

CALL PRINT (.........,0)
    ⋮

CALL TIPE (.........,0)
    ⋮

CALL SYMBØL (.........,0)
    ⋮

CALL TYPFLT (.........,0)
    ⋮

CALL TYPINT (.........,0)
    ⋮

CALL TIPE (0,0,0,0,0,+1)
```

Generally, the module writer should avoid constantly changing plot modes. This is suggested for two reasons. First, some plotters operate very inefficiently in a mode switching environment. Second, should an error exist in either the software or hardware, it might be necessary to dump the generated plot tape. Interpreting this dump would generally be no easy task. However, if the idea of creating subsets of commands was used in generating the plot tape, the task of locating the command(s) causing the problem(s) would be eased considerably in most cases.

All the subroutines used in creating a plot require that at least one of the arguments be the location of a point on the plotting surface. In each case, the point(s) must be specified as real

numbers already scaled to the plotter units. There is no general recommendation as to when this scaling should take place. In some cases, it would be more logical to perform all the scaling at once. In other cases, it would be more logical to perform the scaling on each subset of plot commands. While in other cases, it would be more logical to perform scaling only on an as-needed basis. The choice is left entirely to the discretion of the module writer. Since AXMAX and AYMAX in the /PLTDAT/ named common block define the width and height of the plotting surface in plotter units, and since the plotter origin can always be assumed to be in the lower left corner of the plotting surface, the required scaling ought to be a relatively easy task.

The lower left corner of the plotting surface is always at (0.,0.), while the upper right corner of the plotting surface is always at (AXMAX,AYMAX). These are also the default values of REGIØN (see the /PLTDAT/ named common block) as set by the PLTSET subroutine. The purpose of the REGIØN parameters is to define a rectangular plotting area, outside of which no plotting is to be attempted. For this reason, each of the seven subroutines used in creating a plot always compares the point(s) specified in the calling sequence with the REGIØN values. No portion of any line or axis will be drawn outside the corresponding rectangular plotting area. Nor will any typing be attempted outside this same area. The usage of these REGIØN parameters is left to the discretion of the module writer and the requirements of the module design. In most cases, the default values of REGIØN will not be altered by the module. One situation in which the REGIØN parameters will be altered is when the user has the capability of specifying that a plot is to be drawn only within a specific portion of the total plotting surface and that any part of the plot which appears outside this area is not to be drawn. In any case, if the REGIØN parameters are altered within the module, this can be done at any time on an as-necessary basis.

Having created the desired plot in the plot creation step, the only remaining task for the module writer is terminating the plot. This is a very simple task, accomplished by calling STPLØT:

```
CALL STPLØT (-1)
```

This subroutine, which was also used to initiate the plot, upon sensing a negative argument, will terminate the current subset of plot commands, skip to a new frame (if appropriate), and write an end-of-file mark on the plot tape (if necessary).

If additional plots are to be created within the same module, the entire process just described must be repeated, starting with step 3 (plot initiation). If, however, a new plotter is specified for the succeeding plots, step 2 (parameter initialization) must also be repeated. If a new paper size is specified, subroutine PLTSET must be re-executed prior to repeating step 3.

This concludes the description of the NASTRAN plotting environment. There are two other sections in the NASTRAN Programmer's Manual which deal with this environment. It is suggested that the module writer read these sections also so that he may acquire more of an understanding of the NASTRAN plotting environment than this section affords him. These other sections are: Changes to the Plotter Software, section 6.10.1, and NASTRAN General Purpose Plotter, section 6.10.6.

6.10.6 NASTRAN General Purpose Plotter

One feature which the NASTRAN plotting software lacks is the capability of direct usage of the plotting equipment attached on-line to a computer. This is due not to special purpose programming, but rather to one of the basic characteristics of NASTRAN: computer independence. To access on-line plotters would not only make NASTRAN computer-dependent, but probably installation-dependent also. This installation dependency would result from the necessity of using special subroutines provided by the computer installation to access the on-line plotter, with no guarantee that subroutines having the same name and calling sequences would be available at any other computer installation. Even so, there would almost certainly occur a subroutine naming conflict, due to the great number of subroutines in NASTRAN.

An effort is made in NASTRAN to partially overcome this deficiency. In general, NASTRAN will produce a plot tape which can be used directly by any one of several off-line plotters. In addition, NASTRAN can be directed by the user to produce a so-called "General Purpose Plotter" tape. Another program, completely external to NASTRAN, would then have to exist, its function being to translate this "plot" tape for the on-line plotter so that it will produce the plots intended by NASTRAN. This implies that in order to produce a NASTRAN plot, two programs must be run: first, NASTRAN itself; and then the external translator program.

The purpose of this section is to explain the characteristics and construction of the "NASTRAN General Purpose Plotter" tape, so that a programmer will be able to write a program to translate this "plot" tape for the on-line plotter. Understanding the overall logic used by the NASTRAN plotter software package in producing a plot tape will simplify the task of writing this translator program. It is therefore recommended that the programmer familiarize himself not only with this section of the Programmer's Manual, but also with sections 6.10.1 and 6.10.5, which describe the technique of adding a new plotter to the NASTRAN plotting software package.

The "NASTRAN General Purpose Plotter" tape is composed of a simple set of elementary plot operations, which can easily be deciphered by a FØRTRAN program on any digital computer. As each operation is deciphered, the translator program should direct the on-line plotter to appropriate action. This would normally be done by using the installation software to interface between the translate program and the on-line plotter. With the existence of this external translator program, NASTRAN would then have the capability of indirectly referencing the corresponding on-line

plotter. A by-product of this environment is the implied capability of indirectly accessing any plotter, whether on-line or off-line, assuming the appropriate external translator programs are written.

The "NASTRAN General Purpose Plotter" tape is a seven-track, odd parity, fixed-length record tape. An end-of-file mark follows the last plot only. Each record is composed of 300 six-bit unsigned integers (75 words on an IBM S/360, 50 words on an IBM 7094 or Univac 1108, 30 words on a CDC 6600) and is composed of 10 plot commands, each being composed of 30 six-bit unsigned integers (15 half-words on an IBM S/360, 5 words on an IBM 7094 or Univac 1108, 3 words on a CDC 6600). Not all plot commands will have useful information in all 30 six-bit integers. Some commands use only two of the 30 six-bit integers, while others use 22. The general format of each command is as follows:

$$PCR_4R_3R_2R_1R_0S_4S_3S_2S_1S_0T_4T_3T_2T_1T_0U_4U_3U_2U_1U_000000000 \ ,$$

where:

P = plot command,

C = control index,

R_i = decimal digit of an integer called R,

S_i = decimal digit of an integer called S,

T_i = decimal digit of an integer called T,

U_i = decimal digit of an integer called U,

0 = zero.

The plot command is a six-bit integer, any one of seven (7) possible plot commands, as follows:

0 = no operation,

1 = start new plot,

2 = select camera,

3 = skip to a new frame,

4 = type a character (may also = 14),

5 = draw a line (may also = 15),

6 = draw an axis (may also = 16).

The control index is also a six-bit integer. It may be a pen number, a line density, a camera number, or a pointer into a list of characters and symbols. The four integer values (R,S,T,U) specified in a command must be reconstructed by the external translator program. Each integer value is represented in the command as follows:

$$d_4 d_3 d_2 d_1 d_0 \ ,$$

where the original integer value is given by:

$$d_0 10^4 + d_1 10^3 + d_2 10^2 + d_3 10^1 + d_4 10^0 \ .$$

The significance of each of the four integer values (R,S,T,U) may vary from one plot command to another.

The no-operation (0) command is simply a padding for plot records which may otherwise have been less than 300 characters long. All 30 characters of this command will be zero.

The start-new-plot (1) command will always be the first command introducing each new plot. The first integer (R) will be the plot number. The second and third integers (S and T) are the maximum x and y values specified in any other command for this plot. The minimum x and y values are always zero and are therefore not specified in the start new plot command. If necessary, the translator program can use these maximum x and y values to scale subsequent integer values so that the plot will not exceed the limits of the plotting surface. The plot number is included because some plotters require the plot number as part of the first command for each new plot. In addition, if the receiving plotter is a table plotter, the translator program should issue a command to the plotter which will stop it so that the plotter operator can change the paper. If the plotter is a drum plotter, the translator program must skip a sufficient amount of paper to insure that the previous plot will not be over-plotted. And if the receiving plotter is a microfilm plotter, nothing else need be done.

The select-camera (2) command uses only the control index (C). The remaining 28 characters are always zeros. This command is meaningful only on a microfilm plotter having both film and hardcopy output. The control index is the camera or medium request number: 1 = film only; 2 = hardcopy (paper) only; and 3 = both. Upon receiving this command, the translator program should issue a command to the receiving plotter selecting the requested camera or output medium,

then this command should be ignored.

The skip-to-a-new-frame (3) command also uses only the control index. The remaining 28 characters are always zeros. This command is meaningful only on a microfilm plotter. The control index is the camera or output medium request number: 1 = film only; 2 = hardcopy (paper) only; and 3 = both. The appropriate camera will have already been selected in a previous select-camera command. The only reason the camera number is included in this command is because some microfilm plotters require the camera or output medium to be specified in both a select camera and skip frame command. Upon receiving this command, the translator program should issue a command to the receiving plotter to skip to a new frame. If the receiving plotter is not a microfilm plotter, then this command should be ignored. Note: at least one skip-to-a-new-frame command will appear after each start-new-plot command and before the next start-new-plot command.

The type-character (4), draw-line (5), and draw-axis (6) commands will always occur in sets, i.e., a set of type-character commands, a set of draw-line commands, a set of draw-axis commands. There may be more than one set of each type of command, but within a set the commands will all be of an identical type. This is done because on some plotters it is very inefficient to frequently change modes (e.g., typing mode, line drawing mode) of operation. The plot command of the first command in a set will always = 10 + the basic plot command value, i.e., type-character = 14; draw-line = 15; and draw-axis = 16. In all subsequent plot commands in the set, the plot command value will always equal the basic plot command value.

For a type-character command, the control index is a pointer into a specific list of characters and special symbols. The list of characters to which the pointer applies is Section I of the CHAR94 table (see section 2.5), with the following exceptions: 48 = dot, 49 = circle, 50 = square, 51 = diamond, 52 = triangle (point up). The first two integer values (R and S) in the plot command represent the x and y coordinates of the point on the plotting surface at which the center of the character or symbol should be typed. The remaining 18 characters of the command are always zeros. Upon receipt of a type-character command, the translator program should issue a command to the receiving plotter to type the requested character or special symbol at the specified point. Of course, there is no guarantee that all the possible characters and special symbols can be typed by the receiving plotter. If any character or special symbol cannot be typed by the receiving plotter, the translator program will then have to make a substitution or not type the character at all.

For a draw-line command, the control index is either a pen number (for table and drum plotters) or a line density (for microfilm plotters). If the receiving plotter is a microfilm plotter, it is recommended that the translator program simply draw the line as many times as is indicated by the line density value, rather than using any special density settings available on the plotter hardware. The first two integer values (R and S) represent the x and y coordinates of the starting point of the line. The next two integer values (T and U) represent the x and y coordinates of the ending point of the line. The last 8 characters of the command are always zeros. Upon receipt of this command, the translator program should issue a command to the receiving plotter to draw the line. Note: some plotters require that a line be broken into a series of short lines. If this is the case on the receiving plotter, the translator program will have to accomplish this task unless the installation software makes provision for this automatically.

The draw-axis command is identical to the draw-line command. The only difference is in the orientation of the drawn line. The line drawn by a draw-axis command will always be either horizontal or vertical. For most plotters, the translator program will handle this command just like a draw-line command. However, some plotters which would ordinarily require that lines be broken into a series of short lines, may have a special command available to draw a horizontal or vertical line of any length. For these few plotters only will this command have any special significance in the translator program. If such is the situation, the translator program, upon receipt of this command, should issue a command to the receiving plotter to draw the axis. Otherwise, the translator program should simply issue a command to the receiving plotter to draw a line representing the axis.

6.11 ADDITION OF A NEW LINK

Links can be added to the NASTRAN system with little effort if the total number of links does not exceed fifteen (15). There are at present thirteen (13) links. Links are numbered consecutively, and this should be maintained. If the total number of links must exceed 15, Executive System changes to increase the link limit will be required. These Executive System changes are described in section 6.11.5.

It is assumed that the reader is familiar with the other alternatives for adding new material to the system (section 6.2 through 6.10), and that a new link is normally needed only if the addition of new modules to a present link makes that link non-executable because of an excess of overlay structure or decks. In this case, the programmer generates a new link through the following steps (assuming the link limit is not exceeded):

1. Decide what modules to include in the new link.
2. Add any new modules to the MPL Executive table (see section 2.4.2.2) in deck XMPLBD.
3. Generate a new Link Specification Table and a new link driver.
4. Subsys (create an absolute element of) the new link.

6.11.1 Modules to Include

The following entry points of Executive modules must be included in each new link: XCHK, XCEI, XSAVE, XPURGE, XEQUIV, XSFA and QPARAM. The following non-root segment subroutines must be included in each new link: MSGWRT, USRMSG, BTSTRP, ENDSYS and XEØT. The following DMAP output modules should be included in each new link: TABPT, MATPRN and PRTPARM. Diagrams for the IBM 7094/7040(44) DCS overlay structures for all these routines can be found in section 5.2.9. The addition or use of any other module is at the discretion of the NASTRAN systems programmer. Any existing NASTRAN module can be included in the new link if it is so desired.

6.11.2 Addition of New Modules

All new modules must be added to the MPL Executive table (see section 2.4.2.2). Once this is done, the new MPL needs to be added to the present NASTRAN system before proceeding to generate a new Link Specification Table and a new link driver. The new modules need not be present in the system, but their entry points in the MPL must be in the system.

6.11.3 Generation of a New Link Specification Table and a New Link Driver

The addition of a new link requires that: a) the Link Specification Table, LNKSPC, (see section 2.4.2.7) be updated; and b) a new XSEMi deck (see section 3.3.7), which will be the main program for the new link, be generated.

The FØRTRAN code, in punched card form, necessary to accomplish both of the above tasks can be produced automatically in a NASTRAN run using a LNKSPC update deck as indicated below.

6.11.3.1 Link Specification Table Update

To update the Link Specification Table the user must insert a LNKSPC update deck after the Bulk Data Deck in a NASTRAN run. The processing of a LNKSPC update deck is initiated by logical "sense switches" on a DIAG card in the Executive Control Deck (see explanation below); hence no special header card is required in the LNKSPC update deck. The format of each card of the LNKSPC update deck is as follows:

$$\alpha \quad \beta \quad L_1, \ldots, L_n$$

where

α = Module DMAP name (1 to 8 characters)

β = { Entry point name (1 to 6 characters)
"(NØNE)" (6 characters) if there is no entry point name

$L_1, \ldots, L_n$ = zero or more integers, specifying all links the module resides in.

The α, β and L_i fields must be separated by at least one blank or one comma. An example of a card in a LNKSPC update deck is

```
CHKPNT      XCHK      1,2,3,4,5,6,7,8,9,10,11,12,13,14
```

At present the Executive module CHKPNT with entry point XCHK is in links 1 to 13; the above card will add it to a new link 14. Be sure to include all the other Executive modules in any new link. Cards in this format are repeated until all the modules in the new link have been named. The end of the LNKSPC update is specified by the following card:

```
ENDDATA
```

The following logical sense switches, set by a DIAG card in the Executive Control Deck, must be used to process the LNKSPC update deck:

1. Logical sense switch 29 allows the LNKSPC update deck to be processed.
2. Logical sense switch 28 causes the new Link Specification Table to be punched.*
3. Logical sense switch 31 causes the new Link Specification Table to be printed.

The following DIAG card within the Executive Control Deck

```
DIAG  28,29,31
```

accomplishes this result.

*This code is for the Block Data subprogram XBSBD, and may be compiled under that name as punched, or altered into the present deck to retain the comments presently in that deck. The latter procedure is recommended.

6.11.3.2 Generating New Link Driver Decks

The FØRTRAN code necessary to make a new link driver deck can be produced by setting logical sense switch 30 and the logical sense switches corresponding to the desired link(s). For example the Executive Control Deck card:

```
DIAG 30,2,14
```

will punch the code for XSEM2 and XSEM14. The FØRTRAN code produced, when altered into the deck "XSEMXX" (see section 3.3.8), will produce the desired XSEMi routine. XSEMXX is the model for the link drivers.

This driver code may be produced in the same run as the code for the Link Specification Table, or this coce may be produced by adding the new XBSBD deck to the user's system and then setting the appropriate sense switches on a subsequent run.

If logical sense switch 30 is set, the program will automatically terminate after satisfying all sense switch requests to punch XSEMi's. In essence, the structural problem submitted is only a dummy needed to start NASTRAN. The user is cautioned to check his DIAG card(s) carefully when logical sense switch 30 is set to be sure only the sense switches corresponding to the desired links are set.

6.11.4 Subsys the New Link

The subsys deck necessary to add a new link will vary with the computing machine (see section 5). The safest way for the NASTRAN systems programmer to develop the subsys deck for the new link is to copy the overlay structure of the necessary routines, which are listed in section 6.11.1, and then add the new functional module(s) to this base. The mechanisms for switching from one link to another already exists in NASTRAN up to 15 links, so that, once the subsys of the new link has been accomplished, NASTRAN will be able to assimilate it.

6.11.5 Increasing the Link Limit

To increase the link limit beyond 15, the following Executive System changes must be made.

1. Increase the size of the NAME array in the /SEM/ common block to the desired link limit. Add the additional link names (e.g., NS16, NS17, etc.) to the NAME array via the DATA statement. The /SEM/ common block is defined in the System Block Data subprogram, SEMDBD.

2. Items MAXLNK, DRVRNM, and LNKEDT, defined in subroutine XGPIBS of Executive Preface module XGPI (see section 4.7), must be updated. Increase the value for MAXLNK in the /XLINK/ common block to the desired link limit. Add the additional XSEMi names (e.g., XSEM16, XSEM17, etc.) to the local DRVRNM array via the DATA statement. Add the additional link numbers (e.g., 16, 17, etc.) to the local LNKEDT array via the DATA statement.